UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA INDUSTRIAL ELÉTRICA/AUTOMAÇÃO

ALISSON PERICO
CINDI SAYUMI SHINOHARA
CRISTIANO DELLANI SARMENTO

SISTEMA DE RECONHECIMENTO DE VOZ PARA AUTOMATIZAÇÃO DE UMA PLATAFORMA ELEVATÓRIA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2014

ALISSON PERICO

CINDI SAYUMI SHINOHARA

CRISTIANO DELLANI SARMENTO

SISTEMA DE RECONHECIMENTO DE VOZ PARA AUTOMATIZAÇÃO DE UMA PLATAFORMA ELEVATÓRIA

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do curso de Engenharia Industrial Elétrica – Ênfase em Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Roberto Cesar Betini, Dr. Eng.

CURITIBA

2014

Alisson Perico
Cindi Sayumi Shinohara
Cristiano Dellani Sarmento

Sistema de reconhecimento de voz para automatização de uma plataforma elevatória

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro Eletricista, do curso de Engenharia Industrial Elétrica ênfase Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 13 de fevereiro de 2014.

_____
Prof. Paulo Sérgio Walenia, Esp.
Coordenador de Curso
Engenharia Industrial Elétrica ênfase Automação

_____
Prof. Marcelo de Oliveira Rosa, Dr.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Industrial Elétrica ênfase Automação do DAELT

ORIENTAÇÃO

_____
Prof. Roberto Cesar Betini, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Prof. Roberto Cesar Betini, Dr.
Universidade Tecnológica Federal do Paraná

_____
Prof. Marcelo de Oliveira Rosa, Dr.
Universidade Tecnológica Federal do Paraná

_____
Prof. Winderson Eugenio dos Santos, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Industrial Elétrica ênfase Automação.

RESUMO

PERICO, Alisson; SHINOHARA, Cindi S.; SARMENTO, Cristiano D. **Sistema de reconhecimento de voz para automatização de uma plataforma elevatória.** 2014. 97f. Trabalho de Conclusão de Curso (Graduação - Curso de Engenharia Industrial Elétrica Ênfase em Automação). Universidade Tecnológica Federal do Paraná, 2014.

Este trabalho tem por objetivo apresentar a implementação de um sistema de reconhecimento de voz para automatizar uma plataforma elevatória, importante produto na área da Tecnologia Assistiva. O mercado desta área carece de equipamentos financeiramente acessíveis que sejam adequados às necessidades dos usuários, principalmente no auxílio à mobilidade. O reconhecimento de voz insere-se neste contexto e sua aplicação na Tecnologia Assistiva cresce continuamente. No entanto, em sua maioria, os atuais sistemas disponíveis são caros, dependentes de fontes externas de dados e não possibilitam sua adaptação em aplicações específicas. Por isso, surge a necessidade de se desenvolver um sistema que não apresente essas limitações. Utilizando uma plataforma elevatória disponibilizada pelo PROTA (Programa de Tecnologia Assistiva), o minicomputador *Raspberry Pi* e decodificadores de fala existentes, este trabalho mostrará que é possível desenvolver um sistema de reconhecimento de fala de baixo custo totalmente adaptado à aplicação desejada. Nele são descritas as etapas de desenvolvimento, os testes realizados, os resultados obtidos e as conclusões finais.

Palavras-chave: Reconhecimento de voz. Plataforma elevatória. *Julius*. Projeto Coruja. Tecnologia Assistiva. *Raspberry Pi*. Automação baseada em PC.

ABSTRACT

PERICO, Alisson; SHINOHARA, Cindi S.; SARMENTO, Cristiano D. **Speech recognition system to automate a plataform lift.** 2014. 97f. Trabalho de Conclusão de Curso (Graduação - Curso de Engenharia Industrial Elétrica Ênfase em Automação). Universidade Tecnológica Federal do Paraná, 2014.

The aim of this work is to present the implementation of a system for speech recognition to automate a platform lift, important product in the area of Assistive Technology. The market in this area lacks equipment at affordable prices that are appropriated to the needs of the users, particularly in mobility aids. Speech recognition fits into this context and its application in Assistive Technology is continually growing. However, in most cases, the current available systems are expensive, dependent on external sources of data and do not allow their adaptation to specific applications. Therefore, it is necessary to develop a system that does not present these limitations. Using a lifting platform provided by PROTA (Assistive Technology Program), the Raspberry Pi minicomputer and existing speech decoders, this work will show that is possible to develop a low cost speech recognition system that is fully adapted to the desired application. In this text are described the development stages, the tests performed, the results and the final conclusions.

Keywords: Voice Recognition. Plataform Lift. Julius. Project Coruja. Assistive Technology. Raspberry Pi. PC-based Automation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ADA - *American with Disabilities Act* ou Lei do Americano com Deficiências

API - *Application Programming Interface* ou Interface de Programação de Aplicativos

CAT - Comitê de Ajudas Técnicas

CFTV - Circuito Fechado de Televisão

CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico

COM - *Component Object Model* ou Modelo de Objeto Componente

CORDE - Coordenadoria Nacional para a Integração da Pessoa Portadora de Deficiência

CPU - Unidade Central de Processamento

GPIO - *General Purpose Inputs and Outputs* ou Entradas e Saídas de Uso Geral

GPU - *Graphics Processing Unit* ou Unidade de Processamento Gráfico

HDMI - *High-Definition Multimidia Interface* ou Interface Multimídia de Alta Definição

I^2C - *Inter-Integrated Circuit* ou Circuito Inter-integrado

LaPS - Laboratório de Processamento de Sinais

LCD - *Liquid Crystal Display* ou Display de Cristal Líquido

LXDE - *Lightweight X11 Desktop Environment*

MA - Modelo Acústico

ML - Modelo de Linguagem

MOM - Modelo Oculto de Markov

MP - Unidade de Memória Principal

NOOBS - *New Out Of Box Software*

PC - Computador Pessoal

PIC - *Programmable Interface Controller* ou Controlador de Interface Programável

PIR - *Passive Infrared* ou Passivo Infravermelho

PROTA - Programa de Tecnologia Assistiva

PWM - *Pulse-Width Modulation* ou Modulação por Largura de Pulso

RAM - *Random Access Memory* ou Memória de Acesso Aleatório

RPi - *Raspberry Pi*

SEDH/PR - Secretaria Especial dos Direitos Humanos da Presidência da República

SPI - *Serial Peripheral Interface* ou Interface Periférica Serial

TA - Tecnologia Assistiva

UART - *Universal Asynchronous Receiver/Transmitter* ou Receptor/ Transmissor Universal Assíncrono

UC - Unidade de Controle

UE - Unidade de Entrada

UFPA - Universidade Federal do Pará

ULA - Unidade Lógica e Aritmética

US - Unidade de Saída

SUMÁRIO

1 INTRODUÇÃO

Entende-se como automação qualquer sistema baseado em computador utilizado em indústrias, em serviços ou para o bem estar das pessoas que, substituindo o trabalho humano, reduz os custos e aumenta a qualidade e a segurança dos processos e dos produtos (MORAES e CASTRUCCI, 2007). A automação proporciona uma maior flexibilidade, eficiência e rapidez para a produção, facilitando a execução das mais variadas atividades. Um sistema automatizado é composto basicamente pelas seguintes áreas: sensoriamento; comparação e controle; e atuação (ROCKENBACH, 2004).

A automação inclusiva busca soluções que garantam a acessibilidade, segurança, conforto e saúde de seu usuário (GUEDES et al., 2012). O laboratório do Programa de Tecnologia Assistiva (PROTA) da Universidade Tecnológica Federal do Paraná promove a elaboração deste tipo de tecnologia desenvolvendo recursos e disponibilizando serviços destinados a pessoas com necessidades especiais.

1.1 MOTIVAÇÃO

Segundo Galvão Filho (2009), o princípio da sociedade inclusiva afirma que todas as pessoas portadoras de deficiência devem ter suas necessidades especiais atendidas. Assim sendo, a Tecnologia Assistiva (TA) baseia-se no conceito de inclusão social visando beneficiar o usuário com a melhoria de sua qualidade de vida e com o aumento de sua autonomia. Em nosso país, o acesso à TA ainda é restrito. Os usuários, familiares e profissionais de saúde encontram dificuldades em obter informações sobre as pesquisas em andamento e os produtos disponíveis. Além disso, o mercado nacional carece de equipamentos e o custo deste tipo de tecnologia é elevado (INSTITUTO DE TECNOLOGIA..., 2007).

O CENSO identificou que 23.9% da população brasileira apresenta algum tipo de deficiência[1] e que 7.4% possuem mais de 65 anos (INSTITUTO BRASILEIRO...,

[1] “Foi pesquisada a existência dos tipos de deficiência permanente: visual, auditiva e motora, de acordo com o seu grau de severidade, e, também, mental ou intelectual (INSTITUTO BRASILEIRO..., 2010)”.

2010). O Brasil tem um grande contingente de pessoas que pode se beneficiar da utilização da TA; por isso é importante conhecer as demandas das pessoas com necessidades especiais ou mobilidade reduzida, assim como das idosas, e tornar acessíveis os recursos e os serviços que elas necessitam.

O sistema de automação baseada em computador via comando de voz desenvolvido está inserido nesse contexto, mas não ficará restrito a aplicação em plataformas elevatórias e poderá ser implementado em diversas outras áreas, como, por exemplo, na automação residencial.

Além da aplicação dos conceitos e teorias adquiridos durante o curso de forma integrada, este trabalho ultrapassa os limites acadêmicos, pois considera também o fator humano, uma vez que o objetivo é desenvolver um sistema de TA eficiente e de baixo custo que poderá beneficiar muitas pessoas.

Vale ressaltar que a tecnologia para pessoas sem deficiência visa facilitar o desempenho nas atividades cotidianas. Já para pessoas com deficiência, a tecnologia é caracterizada como uma necessidade, uma vez que torna viáveis atividades consideradas difíceis ou até mesmo impraticáveis (RADABAUGH, 1993).

1.2 OBJETIVOS

1.2.1 Objetivo geral

Automatizar uma plataforma elevatória através do princípio de automação baseada em computador com a implementação de controle via comando de voz. A proposta deste estudo foi aplicada na plataforma elevatória cedida pelo PROTA (Figura 1).

Figura 1 - Plataforma elevatória PROTA
Fonte: Autoria própria.

Além de desenvolver o sistema de comando de voz em *Python*, linguagem compatível com o sistema operacional do *Raspberry Pi,* o *Raspbian*, uma interface com o usuário foi confeccionada a fim de facilitar a visualização de seu estado de funcionamento. A interface é constituída de elementos visuais (LEDs) para indicar ao usuário três possíveis situações: sistema de reconhecimento de voz habilitado, plataforma acionada ou plataforma parada.

1.2.2 Objetivos específicos

- Estudar sobre sistemas de automação baseada em computador, reconhecimento de voz e Tecnologia Assistiva;
- Estudar sobre *softwares* de reconhecimento de voz já existentes: *Julius* e sua variante Coruja;
- Compreender o funcionamento da plataforma elevatória do PROTA;
- Desenvolver o programa de reconhecimento de voz específico para a aplicação desejada;

- Testar o programa desenvolvido;
- Construir placa de interface com o usuário;
- Implementar o sistema de reconhecimento de voz na plataforma elevatória;
- Testar e validar o desempenho do sistema desenvolvido.

1.3 PREMISSA

O mecanismo da plataforma elevatória disponibilizada pelo PROTA foi projetado com base em um sistema de portão eletrônico. O acionamento é realizado somente por controle manual via botoeira ou controle remoto, de tal forma que o sistema de reconhecimento de voz foi instalado paralelamente ao já existente.

As etapas do funcionamento do sistema de reconhecimento de voz proposto são descritas resumidamente a seguir:

Etapa 1: Aquisição do sinal de voz

Um microfone de eletreto é utilizado para a aquisição do sinal de voz do usuário. O sensor de presença PIR (*Passive Infrared* ou Passivo Infravermelho) também é instalado para que o sistema funcione apenas quando uma pessoa estiver sobre a plataforma.

Etapa 2: Processamento do sinal de voz

O sinal de voz do comando dado pelo usuário é transmitido para o computador *Raspberry Pi*. Através do programa de reconhecimento de voz, que utiliza o decodificador *Julius* e um código em *Python* de autoria própria, o processamento do sinal é feito.

O decodificador retorna três parâmetros como resultado: o comando de voz dado em formato de texto, o nível de confiança com que foi interpretado e o coeficiente de Viterbi. Se o resultado do reconhecimento for uma palavra contida no vocabulário definido previamente e se o nível de confiança e o coeficiente de Viterbi forem superiores a valores determinados em função dos testes de validação do sistema, o comando será considerado correto e um sinal para acionamento da plataforma será

enviado. Caso contrário, considera-se que o comando não foi compreendido e o sinal não é enviado.

Etapa 3: Acionamento da plataforma

O sinal para acionamento da plataforma é transmitido do computador através dos dois relés de saída, um para acionamento e outro para parada. Inicialmente, ambos os relés estão em nível baixo (0). Se o relé de acionamento for ativado, sinal nível alto (1), a plataforma se movimentará. Se o relé de parada for acionado, a plataforma irá parar. Se nenhum relé for acionado, o que corresponde à situação de comando não entendido, a plataforma permanecerá no estado atual.

A Figura 2 mostra o fluxograma do sistema a ser desenvolvido. Nota-se que este verifica constantemente a presença, ou não, do indivíduo sobre a plataforma. Uma vez detectada a sua presença, a função de reconhecimento de voz é habilitada. A partir do comando de voz do usuário e a correta interpretação pelo sistema, a plataforma é acionada. Caso contrário, o indivíduo precisa repetir o comando até que ele seja entendido. Após o acionamento da plataforma, o sistema retorna a condição inicial de verificação de presença.

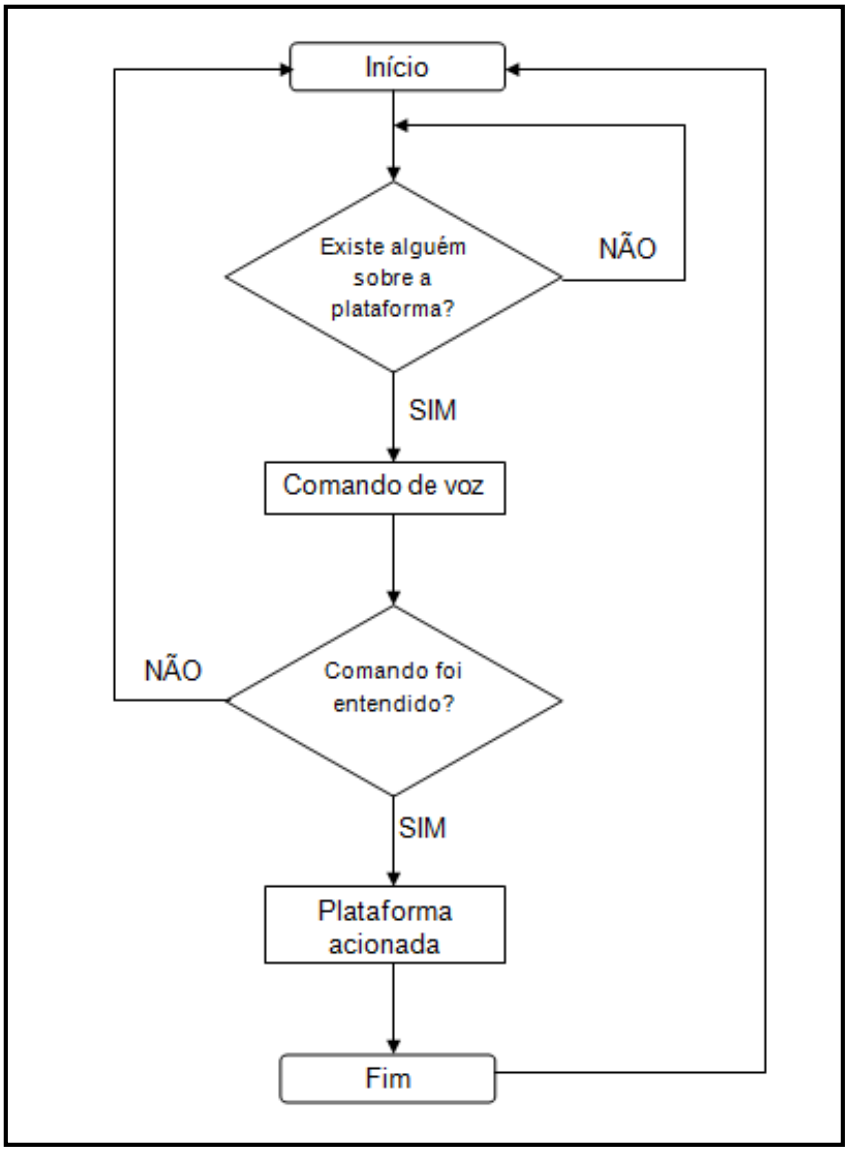


Figura 2 - Fluxograma do sistema de reconhecimento de voz
Fonte: Autoria própria.

1.4 PROCEDIMENTOS METODOLÓGICOS

As etapas descritas abaixo foram organizadas de forma a direcionar o esforço intelectual e técnico da equipe para que os objetivos propostos neste trabalho fossem satisfatoriamente atendidos.

1ª Etapa: Revisão Bibliográfica

Foi levantado material a respeito dos temas abordados em livros, artigos e na Internet sobre os seguintes assuntos:

- Tecnologia Assistiva;
- Automação e domótica;
- Reconhecimento de fala.

2ª Etapa: Análise da plataforma elevatória

Foi analisado o funcionamento da plataforma em seu estado atual e foram estudadas as intervenções necessárias para instalação do programa de reconhecimento de voz.

3ª Etapa: Elaboração do programa de reconhecimento de voz

Utilizando o decodificador *Julius*, um código de programa para a aplicação desejada foi elaborado. Foi necessário estudar a linguagem de programação compatível com o *Raspberry Pi*. Testes de validação foram feitos a fim de verificar a confiabilidade do programa desenvolvido.

4ª Etapa: Implementação do sistema na plataforma e realização de testes

Com o programa finalizado, o sistema foi instalado na plataforma e testes foram realizados para avaliação de sua eficiência. Foi feita a análise dos resultados obtidos e as correções necessárias foram efetuadas.

1.5 ESTRUTURA DO TRABALHO

Este trabalho é composto de cinco capítulos. O primeiro capítulo é destinado à introdução sendo subdividido em motivação, objetivos, premissa e procedimentos metodológicos.

O segundo capítulo explica detalhadamente através de pesquisa bibliográfica os principais conceitos necessários para o bom entendimento deste trabalho: tecnologia assistiva, automação, domótica e reconhecimento de fala.

O terceiro capítulo trata do desenvolvimento do programa de reconhecimento de voz e descreve as etapas referentes à inicialização do minicomputador *Raspberry Pi*, à instalação do *software* base escolhido, à elaboração do código de reconhecimento e ao teste realizado para verificar a eficácia do programa.

O quarto capítulo lista os materiais utilizados para a construção do sistema de reconhecimento de voz e aborda a implementação do sistema de reconhecimento de voz na plataforma elevatória. As diretrizes referentes à metodologia executada são listadas e um estudo sobre a eficiência do sistema de reconhecimento de voz desenvolvido é feito. As dificuldades encontradas com as consequentes correções aplicadas são descritas.

O quinto e último capítulo apresenta as principais conclusões obtidas com o desenvolvimento do trabalho realizado e lista ideias para trabalhos futuros. Nos apêndices podem ser encontrados o código de programa e os arquivos utilizados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão tratados alguns assuntos relevantes para o desenvolvimento do trabalho.

2.1 TECNOLOGIA ASSISTIVA

O termo Tecnologia Assistiva surgiu recentemente e, nos diferentes documentos e publicações referentes ao assunto, pode-se encontrar outras expressões sinônimas como Tecnologias de Apoio e Ajudas Técnicas.

2.1.1 Conceito e objetivos

Em 1988, o conceito foi introduzido oficialmente no ADA (*American with Disabilities Act* ou Lei do Americano com Deficiências), importante conjunto de leis norte-americanas, o qual regulamenta os direitos das pessoas com necessidades especiais nos Estados Unidos. O ADA proporcionou aos norte-americanos o direito de acesso a serviços especializados assim como a recursos necessários a fim de tornar o cidadão mais independente, produtivo e incluído no contexto social (BERSCH, 2008).

Cook e Hussey (1995), baseados no ADA, definiram TA como sendo uma ampla gama de equipamentos, serviços, estratégias e práticas aplicadas para reduzir as dificuldades funcionais encontradas pelos indivíduos com deficiências.

Segundo o Comitê de Ajudas Técnicas (2007), a TA é considerada uma área do conhecimento interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que visam promover a independência, qualidade de vida e inclusão social das pessoas com necessidades especiais ou mobilidade reduzida.

A ISO 9999 utiliza uma definição mais restrita de TA, focada apenas nos recursos. Os produtos, instrumentos, equipamentos ou sistemas tecnológicos, com produção personalizada ou em série, são denominados Ajudas Técnicas e são utilizados para prevenir, compensar, atenuar ou até mesmo eliminar uma deficiência, incapacidade ou desvantagem (ORGANIZAÇÃO INTERNACIONAL..., 2002).

A TA, além de auxiliar pessoas com deficiências físicas, mentais, visuais e/ou auditivas, visa também reduzir ou neutralizar as limitações encontradas por pessoas idosas na execução de suas atividades diárias, aumentando sua autonomia (MINISTÉRIO DA CÊNCIA..., 2005).

Pode-se dizer que o objetivo principal da Tecnologia Assistiva é proporcionar independência e qualidade de vida à pessoa que a utiliza, favorecendo também sua inclusão social, por meio da ampliação da sua mobilidade, controle de ambiente, comunicação e/ou aprendizagem. A TA pode aprimorar uma habilidade funcional deficitária ou possibilitar a realização de uma função desejada que esteja impedida em razão da deficiência ou do envelhecimento (BERSCH, 2008).

2.1.2 Recursos e serviços

A TA baseia-se em dois fundamentos essenciais: recursos e serviços. O ADA define recursos como sendo qualquer item, equipamento ou parte dele, produto ou sistema que visa manter, ampliar ou melhorar as capacidades funcionais de seus utilizadores. Podem ser fabricados em série ou sob medida (ESTADOS UNIDOS, 1988). O nível de complexidade dos recursos varia de acordo com a necessidade do usuário. Uma simples bengala ou um moderno sistema computadorizado podem ser considerados recursos de TA (BERSCH, 2013).

“Serviços são definidos como aqueles que auxiliam diretamente uma pessoa com deficiência a selecionar, comprar ou usar os recursos” (ESTADOS UNIDOS, 1988). É a assistência prestada por profissionais à pessoa com necessidade especial visando fornecer um recurso de TA. Avaliações, prescrições, experimentações, orientações e ensino da utilização do equipamento apropriado são exemplos de serviços. É considerada uma área interdisciplinar que pode envolver profissionais de: Fisioterapia, Terapia Ocupacional, Fonoaudiologia, Educação, Psicologia,

Enfermagem, Medicina, Engenharia, Arquitetura, Design e Técnicos de muitas outras especialidades (BERSCH, 2013).

2.1.3 Tecnologia Assistiva no Brasil

O direito de acesso das pessoas com necessidades especiais ou com mobilidade reduzida aos recursos de TA foi regulamentado pelos Decretos 3.298/1999 e 5.296/2004. A legislação brasileira utiliza o termo Ajudas Técnicas para designar Tecnologia Assistiva.

A Lei nº 10.048/00 refere-se à acessibilidade nos transportes e serviços públicos e a Lei nº 10.098/00 estabelece as normas e os critérios que visam garantir a acessibilidade das pessoas com necessidades especiais ou mobilidade reduzida. Ambas foram regulamentadas pelo Decreto 5.296/2004. Os produtos, instrumentos ou equipamentos adaptados ou especialmente desenvolvidos para ampliar a funcionalidade da pessoa portadora de deficiência e/ou para favorecer a autonomia e independência pessoal são considerados Ajudas Técnicas (BRASIL, 2004).

O Comitê de Ajudas Técnicas (CAT), cuja criação foi prevista no Decreto 5.296, foi instituído em 2006 pela Coordenadoria Nacional para a Integração da Pessoa Portadora de Deficiência (CORDE), um órgão da Secretaria Especial dos Direitos Humanos da Presidência da República (SEDH/PR). O CAT tem como objetivos principais: estruturar as diretrizes e estabelecer as competências da área de Tecnologia Assistiva; realizar estudos visando à elaboração de normas a respeito de Ajudas Técnicas e detectar os centros regionais de referência no assunto, objetivando a formação de uma rede nacional integrada (BRASIL, 2004).

O Portal Nacional de Tecnologia Assistiva, lançado em março de 2006, “é um importante instrumento de convergência e troca de informações sobre as iniciativas existentes no Brasil em pesquisa, desenvolvimento, aplicação e disseminação de TA” (INSTITUTO DE TECNOLOGIA..., 2013). Com o objetivo de qualificar as instituições que contribuem para inclusão social e mapear as competências no Brasil, o Portal subsidia a elaboração de políticas públicas no âmbito da Ciência, Tecnologia e Inovação, sempre buscando novas ideias e soluções para melhorar qualidade de vida das pessoas com necessidades especiais e idosas. O Catálogo de Produtos de

Tecnologia Assistiva reúne informações sobre produtos de TA nacionais ou importados assim como dados de contato das organizações que os comercializam.

2.1.4 Desenho Universal

A definição de Desenho Universal é citada no Decreto 5.296 e é um conceito importante para a construção de uma sociedade inclusiva. A concepção de espaços, artefatos e produtos devem atender simultaneamente as necessidades de todas as pessoas de forma autônoma, segura e confortável (BRASIL, 2004).

O Desenho Universal integra soluções que compõem a acessibilidade. O conceito transcende a ideia de projetos específicos, adaptações e espaços segregados, que respondam apenas a determinadas necessidades, e visa à concepção de ambientes e recursos que contemplem a participação, a utilização e o acesso de todas as pessoas, independente da idade, tamanho, condição física ou sensorial. Por exemplo, é preciso projetar banheiros acessíveis a todas as pessoas desde o inicio do projeto e não ter a necessidade de adaptá-los posteriormente para pessoas com necessidades especiais (GALVÃO FILHO, 2009). Portanto, o conceito de Desenho Universal deve estar cada vez mais presente na formação das Engenharias e Arquitetura (BERSCH, 2008).

2.1.5 Acessibilidade

O Decreto 5.296 também define o termo acessibilidade como sendo as condições para utilização, com segurança e autonomia, por pessoas com necessidades especiais ou mobilidade reduzida dos espaços, mobiliário e equipamentos urbanos; das edificações; dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação.

Segundo Sassaki (2009), a acessibilidade deve estar presente em todos os contextos e aspectos da atividade humana. Para beneficiar a todos, a acessibilidade

necessita ser projetada sob os princípios do Desenho Universal. Considerando a acessibilidade em projetos arquitetônicos, Bersch (2008) a caracteriza como sendo as adaptações estruturais e as reformas em qualquer ambiente que reduzam ou eliminem as barreiras físicas existentes (utilização de elevadores, rampas, sanitários adaptados etc); e os projetos de edificação e urbanismo que garantam acesso, funcionalidade e mobilidade a todas as pessoas.

2.1.6 Plataforma elevatória

A plataforma elevatória é o objeto de estudo neste trabalho e é uma solução muito utilizada para garantir acessibilidade em diversas edificações. Sua instalação tornou-se obrigatória pelo Decreto 5.296/2004, o qual determina que edifícios com mais de um pavimento, além do pavimento de acesso, com exceção das habitações unifamiliares, devem dispor de especificações técnicas e de projeto que facilitem a instalação de equipamento eletromecânico de deslocamento vertical (BRASIL, 2004).

A plataforma elevatória é regulamentada pela ISO 9386-1[2], equivalente no Brasil a NBR 15655-1[3], e pela NBR 9050. Segundo a NBR 9050 (2004), para percurso aberto, a plataforma deve vencer desníveis de até dois metros em edificações de uso público ou coletivo e desníveis de até quatro metros em edificações de uso particular.

2.1.7 Comando de voz no contexto da Tecnologia Assistiva

[2] Norma internacional para plataforma de elevação para pessoas com mobilidade reduzida.
[3] Plataformas de elevação motorizadas para pessoas com mobilidade reduzida - Requisitos para segurança, dimensões e operação funcional/ Parte 1: Plataformas de elevação vertical.

A área que engloba tecnologias baseadas em comando de voz está em constante crescimento. Existe uma grande variedade de sistemas disponíveis no mercado com diferentes níveis de complexidades (DUBRIN, 2002). Além de facilitar o cotidiano das pessoas, cada vez mais é utilizada como auxílio/apoio por pessoas com necessidades especiais ou mobilidade reduzida.

O comando de voz pode ser aplicado nas mais diversas aplicações como, por exemplo: na acessibilidade ao computador (*softwares* de reconhecimento de voz), no controle de ambiente (acionamento por voz dos sistemas de iluminação, áudio e vídeo, ar-condicionado, persianas e diversos outros equipamentos) e na mobilidade do indivíduo (cadeiras de rodas motorizadas e comandadas por voz).

Para a maioria das pessoas, este tipo de tecnologia pode ser considerado meramente um artigo de conforto, mas, para usuários com necessidades especiais, o comando de voz pode facilitar muito a execução de atividades rotineiras (acender uma lâmpada/ abrir uma janela).

2.1.8 Sistemas existentes para controle de plataformas elevatórias via comando de voz

Foi realizada uma busca exaustiva nos sistemas existentes para controle de plataformas elevatórias via comando de voz a fim de situar este trabalho no meio acadêmico/científico e no mercado. Nas seções seguintes, serão apresentados os sistemas encontrados.

2.1.8.1 TCC: Elevador controlado por voz

Os alunos da ETEC Prof. Dra. Doroti Quiomi Kanashiro Toyohara do primeiro semestre de 2013 desenvolveram o trabalho de conclusão de curso intitulado “Elevador controlado por voz”. O objetivo geral era criar um elevador controlado por voz para transporte de diversas cargas e o objetivo específico era de aplicar essa

tecnologia na construção de um projeto visando à acessibilidade de pessoas com necessidades especiais[4] (TEIXEIRA et al., 2013).

A Figura 3 mostra o protótipo de plataforma que os alunos construíram para simular o acionamento via comando de voz. O trabalho foi constituído de três partes básicas: (a) eletropneumática, (b) mecânica e (c) comandos: manual e de voz. A eletropneumática, responsável pelo funcionamento da plataforma, foi composta por um cilindro pneumático de dupla ação, uma válvula 5/3 vias, um compressor de ar e um relé de saída. A mecânica resumia-se a estrutura do protótipo da plataforma elevatória. Para o comando manual, foram utilizados botões de acionamento, saída à relé optoacoplada e um PIC (*Programmable Interface Controller* ou Controlador de Interface Programável) do modelo 16F628A. Já o comando de voz foi constituído por um módulo de voz (microfone) e por uma plataforma de prototipagem Arduino.

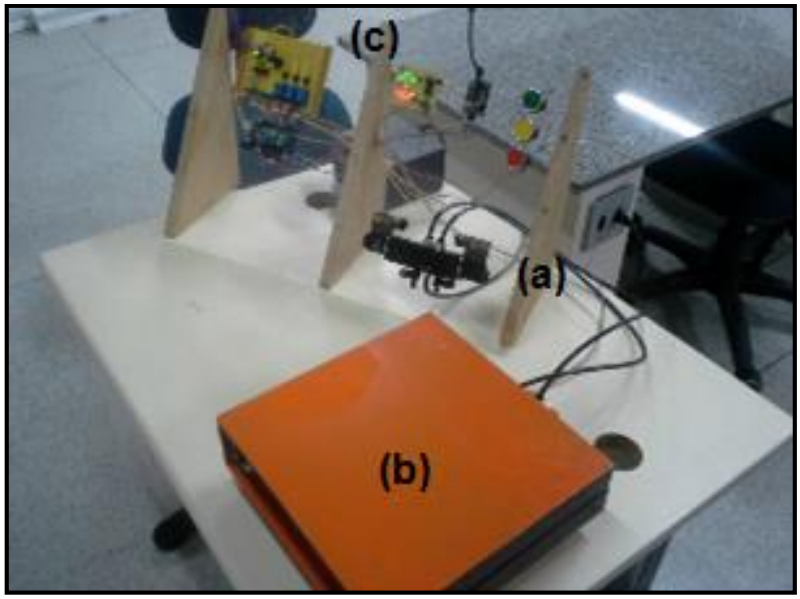


Figura 3 - Protótipo do trabalho de conclusão de curso ETEC:
(a) parte eletropneumática (b) parte mecânica (c) parte de comandos: manual e de voz
Fonte: Teixeira et al. (2013).

O funcionamento da plataforma foi dividido em três etapas: leitura do comando, processamento e saída programada. A leitura do comando, tanto manual (botões) quanto analógico (microfone), fornecia o sinal de entrada do sistema (sobe,

[4] Vídeos sobre o trabalho estão disponibilizados no site: http://ecpv.blogspot.com.br/

desce ou emergência). Após a interpretação desse sinal através do microcontrolador, a saída programada era ativada através de relés optoacoplados. As linguagens de programação utilizadas foram a *Ladder* e a linguagem C.

O investimento necessário para a construção da plataforma elevatória foi relativamente baixo: setecentos e oitenta reais. Na Tabela 1, seguem os custos detalhados.

Tabela 1 - Custos para a construção do sistema utilizado no trabalho de conclusão de curso ETEC

Descrição	Custos
Placa Principal	R$ 51,90
Placa de Sinalização	R$ 38,85
Periféricos	R$72,80
Placa *Palm Switch*	R$ 16,45
Placa de Reconhecimento de voz	R$ 198,00
Parte Mecânica	R$ 410,00
TOTAL	**R$ 780,00**

Fonte: Autoria própria.

O sistema desenvolvido foi aplicado em uma plataforma elevatória (um andar) através da placa apresentada na Figura 4, a qual é composta por saídas a relés, módulo de voz e um PIC.

Figura 4 - Placa para aplicação em plataforma elevatória
Fonte: http://prezi.com/no7b4-uovgob/tcc-elevador-controlador-por-voz/

Uma dificuldade encontrada pelo grupo dos alunos da ETEC está relacionada ao comando de voz e à interpretação do sinal de entrada. Dependendo do tom, da velocidade da voz ou do nível de ruído no ambiente, o comando não é captado. Uma

solução a ser aplicada consiste no aprimoramento da programação, utilizando um banco de dados com diversas gravações nos mais variados ambientes e com diferentes locutores. Vale frisar que o trabalho dos alunos da ETEC era um protótipo, não um sistema funcional.

2.1.8.2 Elevadores

Os sistemas de voz também podem ser encontrados em elevadores. Os mais utilizados servem apenas para informar o andar em que a cabina está situada ou o sentido de deslocamento da mesma. No entanto, existem elevadores que realmente funcionam por comando de voz. Estes apresentam uma demanda pequena e um custo elevado quando comparado com elevadores convencionais.

O elevador desenvolvido pela Mitsubishi Electric em 2011 é composto por um sistema de reconhecimento de voz e por um sensor de presença. Baseado nas tecnologias de navegação utilizados nos veículos automotivos, o elevador é capaz de reconhecer o comando de qualquer voz, sem treinamento prévio.

A Figura 5 mostra o princípio de funcionamento do elevador da Mitsubishi. O elevador é chamado se detecta a presença de uma pessoa aguardando em frente à porta (a). Uma vez em seu interior, o usuário fala o andar desejado e aguarda a confirmação sonora fornecida pelo sistema de que o comando foi bem compreendido (b). Após isso, o elevador se põe em movimento e alerta quando o destino final for alcançado. Essa interação com o usuário foi desenvolvida para atender as necessidades de pessoas com deficiência visual ou com mobilidade reduzida.

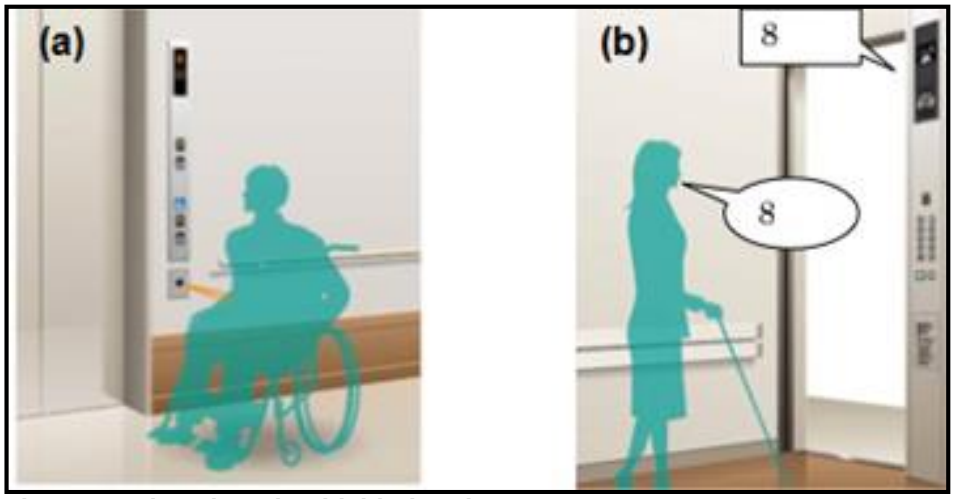


Figura 5 - Elevador Mitsubishi Electric
(a) Detecção da presença (b) Compreensão do comando de voz
Fonte: Adaptado do site Mitsubishi Electric.

No final de 2006, o primeiro elevador brasileiro controlado por comando de voz chegou ao mercado. O protótipo desenvolvido pela Engetax Elevadores foi instalado em caráter experimental no Centro de Negócios de São Paulo. Seu diferencial era que poderia ser chamado por intermédio de um aparelho *Palm Top*, pelo celular ou por microfone, facilitando seu acionamento (CONDOWORKS apud. Gazeta do Povo, 2006).

Atualmente, a Engetax Elevadores disponibiliza aos seus clientes um pacote adicional de sistema de comando de voz para plataformas elevatórias. O custo de instalação é estimado em dez mil reais[5].

2.2 AUTOMAÇÃO E DOMÓTICA

A automação difundiu-se amplamente no setor industrial ao longo de sua história com o objetivo de substituir a mão-de-obra humana e, consequentemente, aumentar os ganhos da produção do estabelecimento. Para seu desenvolvimento, foi necessária a criação de *hardwares* e *softwares* cada vez mais modernos e robustos.

[5] A empresa foi contatada por integrantes da equipe e um orçamento foi feito (plataforma elevatória de uso residencial para até dois metros de deslocamento vertical). A empresa alegou não possuir material de divulgação para utilização neste trabalho, pois a procura por este tipo de tecnologia é muito baixa.

Segundo Pinheiro (2004), automação é a capacidade de se executar comandos, obter medidas, regular parâmetros e controlar funções automaticamente, sem a intervenção humana.

Esse conceito passou a fazer parte também dos projetos residenciais, utilizando-se das tecnologias já elaboradas pela automação industrial para automatizar processos dentro de uma unidade residencial e, com isso, propiciar benefícios em setores como: gerenciamento de recursos, economia, conforto, segurança dos usuários e prevenção de acidentes.

De acordo com Chan et al. (2008), a automação de residências é uma das áreas mais promissoras para o desenvolvimento de tecnologias eletrônicas. O sistema domótico integra uma série de equipamentos de sensoriamento - a fim de obter informações sobre o meio em que se encontra - com um sistema de controle, que efetua ações para supervisionar e gerenciar os dispositivos.

2.2.1 Sistemas de automação baseados em PC

Computadores pessoais (PCs) são populares, poderosos e, atualmente, estão sendo amplamente utilizados para cálculos, comunicação, processamento de texto, de dados e outros trabalhos. Eles também são usados para a medição precisa, aplicações de controles em indústrias e instituições de pesquisa e permitem a fácil expansão e adoção de qualquer aplicativo desejado (MATHIVANAN, 2006).

A arquitetura básica de um computador ainda segue os conceitos estabelecidos por John Von Neumann (1903-1957), o qual a define como sendo máquinas que codificam instruções que podem ser armazenadas na memória e processadas posteriormente.

Segundo Leite (2006), o computador, independentemente do seu porte, possui quatro unidades básicas integradas: Unidade de Entrada (UE), Unidade Central de Processamento (CPU), Unidade de Memória Principal (MP) e Unidade de Saída (US), que se integram como mostrado na Figura 6.

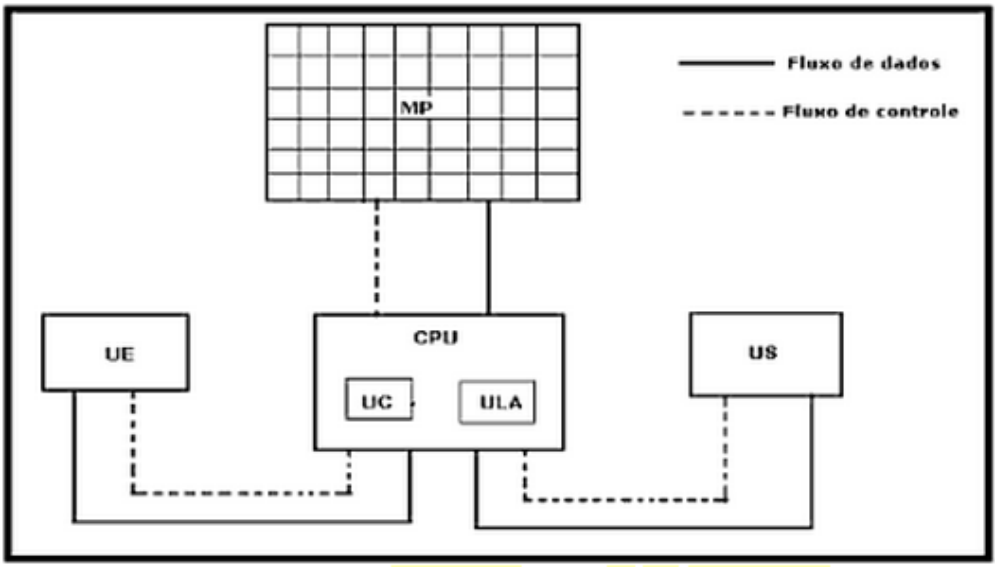


Figura 6 - Esquema da arquitetura básica de um computador
Fonte: Leite (2006).

A UE é responsável pela entrada de dados no computador. Já a US realiza a tarefa inversa da UE, ela decodifica os sinais eletrônicos gerados pelo processamento em informações para o usuário. A CPU é a unidade mais importante do computador, é ela que processa os dados e as instruções de um programa. A MP tem a função de armazenar temporariamente os dados e instruções para posterior processamento pela CPU. Essa capacidade de armazenamento permite o processamento automático, segundo o conceito de Von Neumann.

A CPU é composta pela Unidade de Controle (UC), que é responsável pelo controle das ações a serem realizadas pelo computador, Unidade Lógica e Aritimética (ULA), a qual executa as principais operações lógicas e aritméticas do computador e por registradores, uma unidade de memória capaz de armazenar bits.

2.2.1.1 *Raspberry Pi*

O *Raspberry Pi* é um computador do tamanho de um cartão de crédito desenvolvido no Reino Unido pela Fundação *Raspberry Pi*. Este pequeno equipamento surgiu em 2006 e tem sido aprimorado desde então. Possui todos os

elementos centrais de um PC e tem como objetivo disponibilizar um computador simples e de baixíssimo custo.

Entre os componentes do *Raspberry Pi* destaca-se um pequeno circuito integrado que reúne o processador com a arquitetura ARM (arquitetura de processador de 32 bits), um GPU (*Graphics Processing Unit*, ou Unidade de Processamento Gráfico) VideoCore IV e a memória RAM (*Random Access Memory* ou Memória de acesso aleatório).

O *Raspberry Pi* vêm com 26 pinos GPIO (*General Purpose Inputs and Outputs* ou Entradas e Saídas de Uso Geral), os quais são pinos programáveis para entrada e saída de dados, onde é possível ligar os mais diversos equipamentos. Não possui armazenamento em massa interna, por isso requer um cartão SD de memória *flash*, normalmente usado em câmeras digitais, configurado de tal forma a se comportar como um disco rígido para o processador do RPi. O RPi vai carregar o sistema operacional na memória RAM a partir deste cartão, da mesma forma como um PC carrega um sistema operacional a partir do seu disco rígido (*RASPBERRY PI...*, 2012). A Figura 7 ilustra um *Raspberry Pi*.

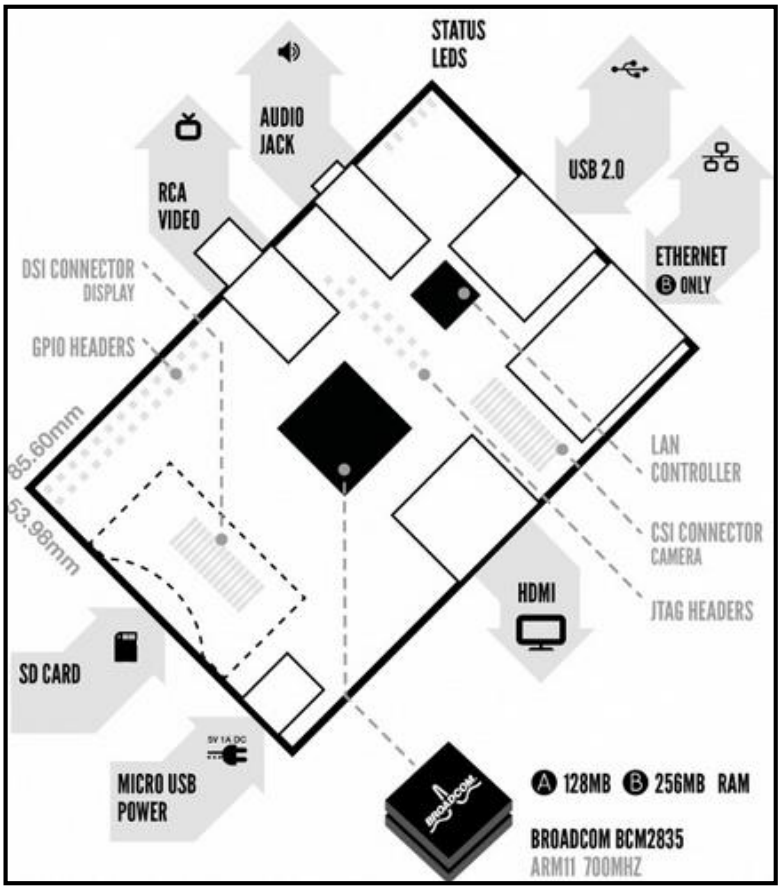


Figura 7 - *Raspberry Pi*
Fonte: Adaptado de *Raspberry Pi Foundation* (2012).

O *Raspberry Pi* (RPi) ainda dispõe de porta USB para comunicação com periféricos, suporte para três saídas de vídeos diferentes: Vídeo composto, HDMI, que também carrega sinal de áudio, e vídeo DSI. Disponibiliza também conector para áudio e conector de rede *Ethernet*.

A Tabela 2 compara os dois modelos de *Raspberry Pi* existentes, A e B.

Tabela 2 - Comparação entre modelos *Raspberry Pi* A e B

Raspberry Pi	**Modelo A**	**Modelo B**
Ethernet	-	onboard 10/100 *Ethernet* RJ45 jack
Conector USB	Um	Dois
Memória	256MB SDRAM	512MB SDRAM
Processador de aplicações multimídia	Broadcom BCM2835 SoC full HD	Broadcom BCM2835 SoC full HD
CPU	700 MHz Low Power ARM1176JZ-F Applications Processor	700 MHz Low Power ARM1176JZ-F Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor	Dual Core VideoCore IV® Multimedia Co-Processor
USB 2.0	Um conector USB	Dois Conectores USB
Saída de Vídeo	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)
Saída de Audio	3.5mm jack, HDMI	3.5mm jack, HDMI
Armazenamento	SD, MMC, SDIO card slot	SD, MMC, SDIO card slot
Sistema Operacional	Linux	Linux
Dimensão	8.6cm x 5.4cm x 1.5cm	8.6cm x 5.4cm x 1.7cm
Alimentação	5 Volts	5 Volts
Preço (aquisição no Brasil)	R$ 170,00	R$ 250,00

Fonte: Adaptado de http://downloads.element14.com/raspberryPi1.html

Levando em conta todas as características esperadas do processador de sinal do reconhecedor de fala desenvolvido, concluiu-se que uma alternativa de baixo custo dentre os reconhecedores de fala disponíveis no mercado é o computador *Raspberry Pi*. Dentre suas principais características, destacam-se para o presente trabalho:

- Baixo custo;
- Entradas e saídas digitais para uso geral (16 interfaces configuráveis);
- Comunicação E*thernet*;
- Alta capacidade de processamento (700 MHz);
- Ampla gama de linguagens de programação disponíveis.

2.2.1.1.1 Funcionamento GPIO

O *Raspberry Pi* possui 26 pinos GPIO, que podem ser configurados para proporcionar diversas funções:

- Serial *UART* (*Universal Asynchronous Receiver/Transmitter* ou Recepto/Transmissor Universal Assíncrono), um módulo que contém, ao mesmo tempo, os circuitos de transmissão e recepção necessários para as comunicações seriais assíncronas;
- SPI (*Serial Peripheral Interface* ou Interface Periférica Serial), um protocolo de dados seriais síncronos utilizado para comunicação entre controlador e um ou mais periféricos que também pode ser utilizado entre dois controladores;
- PWM (*Pulse-Width Modulation* ou Modulação por Largura de Pulso), a modulação da razão cíclica (*duty cycle*) de um sinal para transportar qualquer informação sobre um canal de comunicação ou controlar o valor da alimentação entregue à carga;
- I²C (*Inter-Integrated Circuit* ou Circuito Inter-integrado), um barramento de comunicação criada pela Philips, para chips para se comunicarem uns com os outros;
- Entrada e saída digitais;
- Pinos de alimentação 3,3V e 5V.

O uso de qualquer um dos pinos exige cuidado, pois eles possuem nível lógico de 3,3V e não são tolerantes a níveis de 5V, já que são ligados diretamente ao *Broadcom chip* no coração do *Raspberry Pi*; isso significa que não há proteções.

Para a programação e utilização dos pinos GPIO foi instalada uma biblioteca *Python* que diminui a complexidade de controle dos pinos GPIO, através do comando:

```
apt-get install python-rpi.gpio
```

Com a biblioteca instalada, no código *Python* utiliza-se a linha de código abaixo para importá-la:

```
import RPi.GPIO as GPIO
```

Então se define a numeração dos pinos GPIO da placa. Neste caso há duas possibilidades, BOARD à esquerda e BCM à direita de acordo com a Figura 8 e da linha de código a seguir:

```
GPIO.setmode(GPIO.BCM)
```

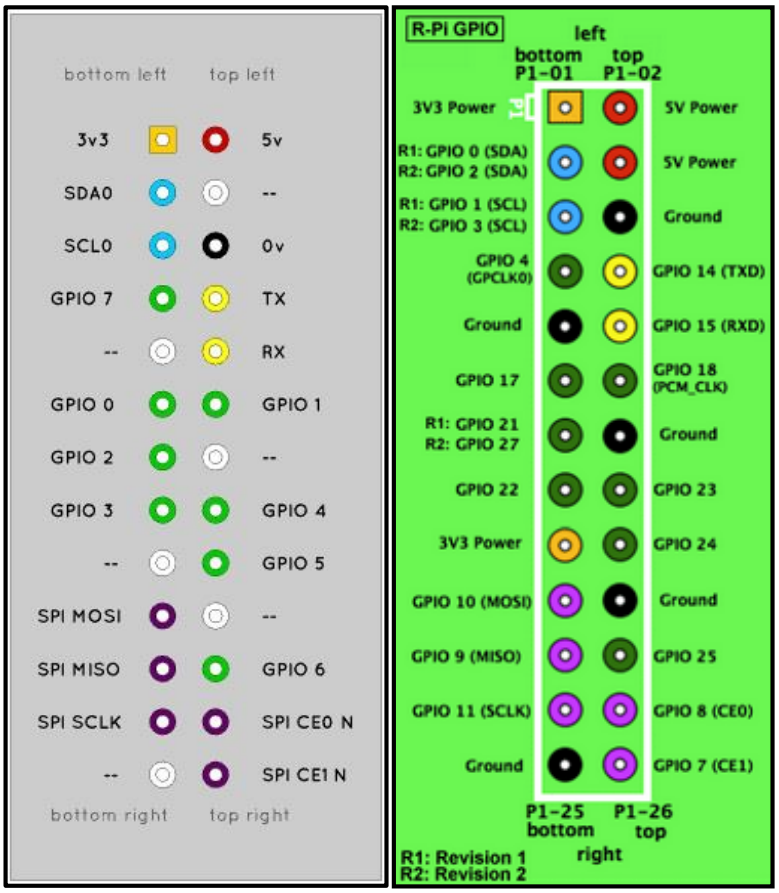


Figura 8 - Diagrama de pinos GPIO
Fonte: http://openmicros.org/index.php/articles/94-ciseco-product-documentation/raspberry-pi/217-getting-started-with-raspberry-pi-gpio-and-python

BOARD significa usar os números físicos dos pinos do conector do *Raspberry Pi*, BCM significa a designação pelo número do *Broadcom SOC*.

Na sequência do código, configura-se cada pino como entrada (*IN*) ou saída (*OUT*) conforme abaixo:

```
GPIO.setup(4, GPIO.IN)

GPIO.setup(27, GPIO.OUT)
```

Com todas as configurações iniciais feitas, é possível controlar os pinos de saídas com os comandos para ligar ou desligar, respectivamente com as seguintes linhas de código:

```
GPIO.output(27, True)

GPIO.output(27, False)
```

A leitura de um pino de entrada é feita pela linha de código abaixo, que retornará True ou False:

```
GPIO.input(25)
```

2.2.1.1.2 *Python*

O *Python* é a linguagem de programação orientada a objetos disponível no *Raspberry Pi* para desenvolvimento de aplicações. Com sintaxe simples, é compatível com os seguintes sistemas operacionais: Windows, Linux/Unix e Mac OS X. Pode ser utilizado também em Java e em máquinas virtuais .NET. Administrado pela *Python Software Foundation*, é um recurso com código aberto que pode ser usado gratuitamente até mesmo no âmbito comercial (*PYTHON SOFTWARE*..., 2013).

Esta linguagem dinâmica, rápida e acessível, disponibiliza recursos avançados de manipulação de estruturas de dados e eficientes mecanismos para detecção e tratamento de erros. Com tamanho reduzido quando comparado a códigos

escritos em Java ou C++ (um quinto a um terço do tamanho), o código (*script*) não necessita de compilação previa para execução. As bibliotecas possuem variados conjuntos de funcionalidades e suportam diversas tarefas de programação como, por exemplo: manipulação de *string*, *math*, *time* e GPIO (ASCHER; LUTZ, 2004).

A programação em *Python* é facilitada por existirem vários fóruns de discussões na Internet, nos quais é possível encontrar informações e tutoriais sobre a estrutura da linguagem, funções e bibliotecas disponíveis.

A versão utilizada para o desenvolvimento deste trabalho foi: *Python* 2.7.3. Os códigos foram escritos no editor de texto *Leafpad* e executados no terminal do *Raspberry Pi* através do comando *python nome_do_arquivo.py*[6].

2.2.2 Sistemas operacionais

O sistema operacional forma uma interface entre o usuário e o *hardware*. O sistema operacional, juntamente com alguns outros *softwares*, pode ser agrupado como *softwares* do sistema, tais como editores, compiladores, utilitários, sistema operacional e aplicativos (SRIRENGAN, 2006).

Flynn e Mchoes (2002) comparam o sistema operacional a uma espécie de gerente executivo, uma parte do sistema de computação que administra todos os componentes de *hardware* e *software*. Em outros termos, o sistema operacional controla cada arquivo, dispositivo, seção de memória principal e instante de tempo de processamento. Os principais e mais usados sistemas operacionais são: Windows, Mac OS e Linux.

2.2.2.1 Linux

[6] Lembrando para execução, é necessário estar no diretório onde foram salvos os arquivos de texto.

O sistema operacional Linux, pode ser definido como: um sistema operacional de livre distribuição, semelhante ao Unix, constituído por um *kernel* (componente central do sistema operacional), ferramentas de sistema, aplicativos e um completo ambiente de desenvolvimento (DELGADO, 2005).

De acordo com Delgado (2005), por ser um sistema operacional livre, é resultado do trabalho de milhares de colaboradores, universidades, empresas de *software* e distribuidores ao redor do mundo. Como seu desenvolvimento é feito sob o projeto GNU (sistema operacional baseado em *software* livre) da *Free Software Foundation,* qualquer pessoa pode fazer alterações em seu código fonte, desde que este seja distribuído.

Segundo Campos (2006), o Linux é ao mesmo tempo um *kernel* (ou núcleo) e o sistema operacional que roda sobre ele. O *kernel* Linux foi criado por Linus Torvalds, desenvolvedor reconhecido mundialmente e integrante mais representativo da *Linux Foundation*, em 1991, e hoje é mantido por uma comunidade mundial de desenvolvedores, dos quais é possível citar empresas como HP, IBM, Hitachi.

Segundo Delgado (2005), algumas características importantes presentes nos sistemas operacionais modernos também presentes no Linux são:

- Multiplataforma: O Linux opera em computadores de outras famílias e compatíveis;
- Multiprocessado: Possui suporte a computadores com mais de um processador;
- Multitarefa: Execução de vários programas ao mesmo tempo;
- Multiusuário: Usuários podem operar a mesma máquina ao mesmo tempo.

2.2.2.1.1 *Raspbian*

O *Raspbian* é um sistema operacional livre baseado em *Debian* GNU Linux otimizado para o hardware *Raspberry Pi*. Possui um conjunto de programas básicos

e utilitários que possibilitam o funcionamento do *Raspberry Pi*. O sistema disponibiliza 35 mil pacotes de *software* pré-compilado de fácil instalação no *Raspberry Pi.* A construção inicial desses pacotes foi concluída em junho de 2012, contudo, o *Raspbian* ainda está em constante desenvolvimento, visando à melhoria da estabilidade e desempenho.

O *Raspbian* é uma versão não oficial do *Debian* com configurações de compilação ajustadas para produzir código otimizado que é executado no *Raspberry Pi*. Isto proporciona um desempenho significativamente mais rápido para aplicativos que fazem uso intenso de operações aritméticas em ponto flutuante. Todas as outras aplicações também ganham algum desempenho através do uso de instruções avançadas da CPU do *Raspberry Pi* (*RASPBIAN*, 2014).

Trazendo o ambiente *Lightweight X11 Desktop Environment* (LXDE), um ambiente de *desktop* de código aberto para *Unix*, e outras plataformas como *Linux*, o *Raspbian* coloca um ambiente *desktop* completo no *Raspberry Pi*. O sistema contém basicamente o ambiente *desktop LXDE*, o gerenciador de janelas *Openbox*, o navegador *Midori*, ferramentas de desenvolvimento de *software* (*Leafpad*) e código fonte de exemplo para funções multimídia.

2.2.3 Sensores e transdutores

O termo sensor é definido como um dispositivo que é sensível a um fenômeno físico e transmite um sinal elétrico para um dispositivo de medição ou controle, portanto sensores são transdutores (BOLTON, 2008).

Um sensor tem por objetivo detectar um evento e enviar sinais elétricos analógicos ou digitais, relativos às grandezas monitoradas que serão reconhecidos por um equipamento controlador (BOLZANI, 2004).

No âmbito deste trabalho, serão utilizados os seguintes sensores: de presença (PIR), de voz (microfone) e de proximidade (*reed switch*), cujos funcionamentos serão detalhados a seguir.

2.2.3.1 Sensor Infravermelho Passivo (PIR)

O sensor de movimento PIR é um sensor eletrônico, simples de usar, que mede radiação infravermelha de objetos, que está diretamente relacionada à temperatura do objeto, em seu campo de visão. Todos os objetos com uma temperatura acima do zero absoluto emitem energia térmica sob a forma de radiação. Normalmente, esta radiação é invisível ao olho humano, mas pode ser detectado por dispositivos eletrônicos concebidos para esse fim.

Por ser um dispositivo passivo, o sensor não transmite um sinal, em vez disso, ele responde às alterações na radiação do campo de visão do sensor causada por um objeto. O padrão de detecção esta subdividido em segmentos angulares. A detecção de movimento de uma fonte de calor se dá quando esta fonte se move de um segmento para outro, gerando um sinal elétrico.

O princípio básico de funcionamento do sensor PIR, segundo Garcia (2005), depende da diferença de energia infravermelha entre um intruso e o fundo. O sensor tem duas ranhuras, cada ranhura é feita de um material especial que é sensível ao infravermelho; assim ele detecta a mudança na radiação infravermelha quando seu campo de visão é bloqueado por um objeto que tem temperatura diferente do fundo, como mostra a Figura 9.

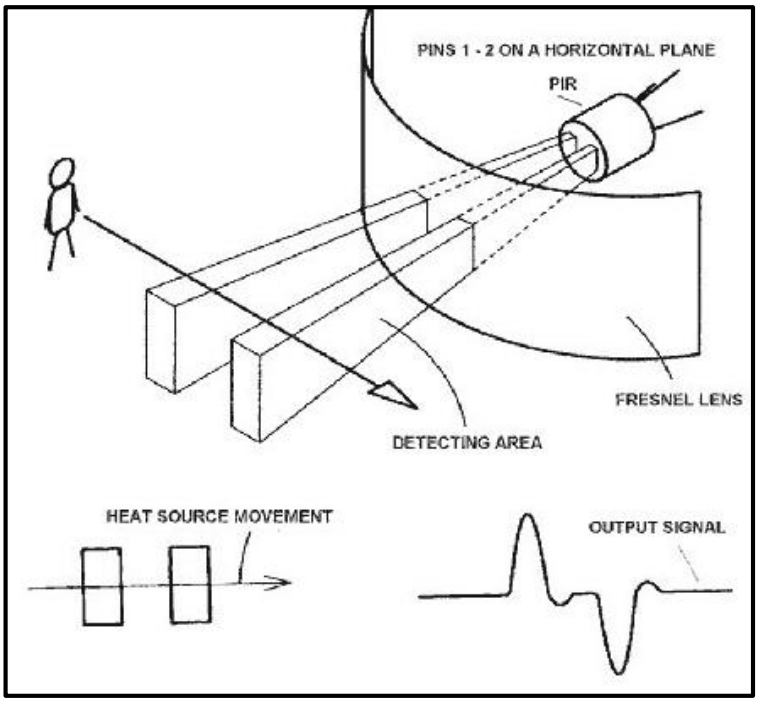


Figura 9 - Detecção PIR
Fonte: http://www.ladyada.net/learn/sensors/pir.html

A probabilidade de detecção do sensor PIR é uma função da amplitude da diferença entre a temperatura do intruso e o fundo, mas também depende de outros fatores, tal como a velocidade em que um diferencial de temperatura é introduzido na zona de detecção do sensor.

O sensor requer uma tensão de alimentação de 5V a 12V. O sensor PIR emite um pulso digital alto de 3,3 V quando é acionado e o pulso baixo é emitido quando não há detecção de movimento. Possui uma sensibilidade de até 6 metros e uma faixa de detecção de 110°. Os comprimentos de pulso e a sensibilidade podem ser alterados pelos *trimpot* que acompanham a placa de circuito impresso do sensor PIR.

O sinal desse sensor terá o papel de habilitar o reconhecimento de voz, sendo assim, o reconhecimento de voz irá iniciar somente após a detecção de um usuário na plataforma, evitando a interpretação de comandos falsos.

2.2.3.2 Microfone

Patsko (2006) define microfone como sendo um componente captador de som, destinado a converter as vibrações sonoras em sinais elétricos. Eles são empregados em inúmeras aplicações, desde aparelhos telefônicos e equipamentos de gravação à sensores de distância baseados em ultrassom.

O microfone adequado para aplicações de comando de voz é o microfone de eletreto, ilustrado na Figura 10. Ele é um modelo muito comum, que possui um preço acessível e fácil utilização, pode ser encontrado em diversos aparelhos, até mesmo equipamentos que necessitam de um microfone para gravações de alta qualidade.

Figura 10 - Microfones de eletreto
Fonte: Patsko (2006, p. 25).

De acordo com Patsko (2006), o sinal de tensão gerado pelos microfones de eletreto se dá pela vibração de um diafragma, devido incidência de ondas sonoras sobre ela, juntamente com uma pequena placa de metal fixa que compõem um capacitor. A vibração do diafragma faz com que a distância entre ele e a placa de metal varie e, consequentemente, ocorre à variação da capacitância desse conjunto. Como a carga armazenada pelo capacitor se mantém constante, tem-se também a variação da tensão entre o diafragma e a placa de metal.

Com a finalidade de captar os comandos de voz, é utilizado o adaptador de áudio 3,5mm para USB A-5572A da Logitech, através do qual é conectado o microfone condensador *Leadership* ao *Raspberry Pi*.

2.2.3.3 *Reed Switch*

O *reed switch* consiste numa ampola de vidro no interior da qual existem duas lâminas flexíveis com contatos especiais em suas extremidades como mostra a Figura 11.

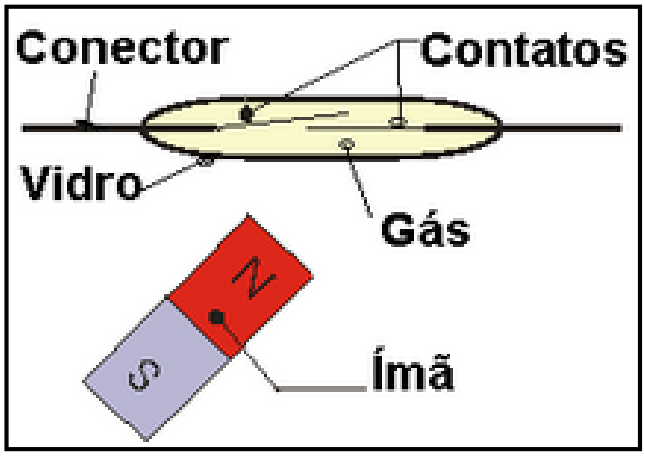


Figura 11 - *Reed Switch*
Fonte: http://www.if.ufrgs.br/mpef/mef004/20061/Cesar/SENSORES-Reed-switch.html

A ampola, para evitar a oxidação dos contatos, é cheia com um gás inerte. As lâminas sobrepõem-se no interior da cápsula de vidro com um espaço entre eles, apenas fazendo contato um com o outro na presença de um campo magnético adequado. As pás de palheta agem como condutores de fluxo magnético, quando exposto a um campo magnético externo, a partir de um imã permanente ou de uma bobina eletromagnética, polos de polaridade oposta são criados e os contatos se fecham quando a força magnética excede a força das lâminas. À medida que o campo magnético externo for reduzido, de modo que a força entre as palhetas é menor do que a força de restabelecimento das pás de palheta, os contatos abrem (TSCHULENA et al., 2003).

O *reed switch* funciona como um interruptor de lâminas com a ação de um campo magnético e serve como fim-de-curso para a plataforma elevatória. Estão presentes dois r*eed switches* normalmente abertos (5 V), uma na parte inferior (chão) e outro na parte superior (primeiro andar). Quando a plataforma se aproxima dos sensores, o campo magnético faz com que as lâminas se magnetizem e curvam-se de modo a encostar o contato comum ao contato NA (normalmente aberto). A corrente pode então circular entre estes dois terminais e enviar sinal para parar a plataforma (0 V).

2.3 RECONHECIMENTO DE FALA

A tarefa automática de reconhecimento da fala humana se resume a gerar uma representação digital de um sinal acústico, para que um processador de dados possa trabalhar com esta informação. O crescente interesse em se desenvolver sistemas capazes de realizar operações comandadas pela fala se deve muito ao fato de este ser o meio de comunicação humana mais fundamental (COSTA, 2008).

Silva (2009) divide a classificação de complexidade de um reconhecedor de fala em três categorias:

- Dependência de locutor – Um reconhecedor dependente de locutor é aquele que tem seu banco de dados gravado por uma pessoa apenas e, portanto, interpretará os sinais acústicos provenientes somente desta pessoa. Já um independente de locutor possui um vasto e diversificado banco de dados para poder identificar qualquer locutor;
- Tamanho do vocabulário – Um reconhecedor de até vinte palavras possui um vocabulário pequeno; até cem palavras, médio; até mil palavras, grande; e acima de mil palavras, muito grande;
- Modo de pronúncia – Um reconhecedor de palavra isolada necessita detectar um início e um fim para cada palavra pronunciada, enquanto um de palavras conectadas consegue reconhecer frases inteiras, desde que cada palavra seja pronunciada claramente. Finalmente, o modelo mais complexo existente é capaz de interpretar a fala contínua, sem restrições quanto a diferentes dialetos, pronúncias e ritmos de fala.

Dentro da classificação de complexidade apresentada, pode-se dizer que o reconhecedor almejado deverá ser independente de locutor, possuir um vocabulário pequeno e interpretar palavras isoladas.

Outra característica que vale destaque com relação aos reconhecedores de fala é a dificuldade em se encontrar equipamentos para uso específico, como é o caso do presente trabalho, que sejam financeiramente acessíveis. Sendo assim, este tópico tem por objetivo expor os conhecimentos técnicos necessários no desenvolvimento de um sistema simples de reconhecimento de fala, de baixo custo.

2.3.1 Sinal acústico da fala

No processo de desenvolvimento de um sistema de análise de sinal acústico da fala, é preciso identificar alguns padrões desse sinal. Sendo assim, na sequência serão apontados alguns dados referentes a características típicas do sinal de voz.

Plannerer (2005) destaca algumas características importantes do sinal da fala:

- A largura de banda do sinal é de 4 kHz;
- O sinal é periódico com uma frequência fundamental entre 80 e 350 Hz.

Picone (1996) cita, também, algumas características que valem destaque:

- Três frequências ressonantes são normalmente encontradas na largura de banda de 4 kHz;
- Alguns sons, como o do “s”, possuem altíssimas larguras de banda.

A Figura 12 mostra um exemplo de como a intensidade de energia é distribuída de acordo com a frequência, em valores médios amostrados entre homens e mulheres.

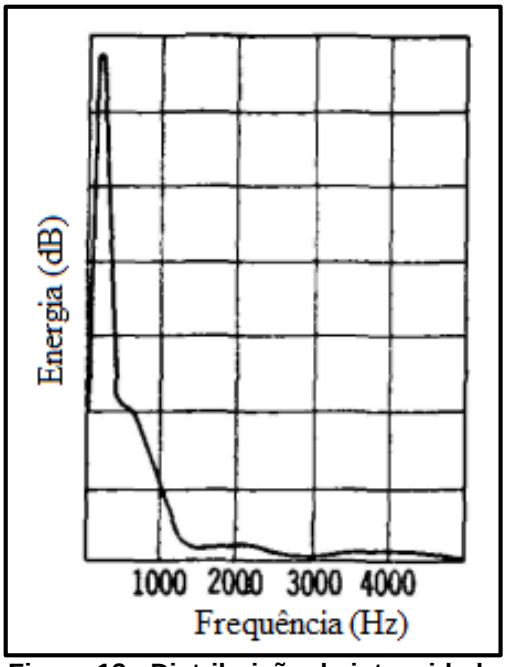


Figura 12 - Distribuição da intensidade de energia da fala de acordo com a frequência
Fonte: Adaptado de http://www.vias.org/albert_ecomm/aec02_acoustic_speech_power.html

Picone (1996) afirma que, devido ao formato do trato vocal, que pode ser aproximado por um tubo fechado de 17 cm, ocorrem picos no espectro de frequência do sinal da fala em múltiplos ímpares de 500 Hz. Esses picos levam o nome de frequências formantes.

Na Figura 13 são exemplificados os sinais analógicos de três fonemas (“i”, “e” e “é”) e suas respectivas distribuições de frequência. Na coluna (a), representam-se as posições de cada parte da boca ao pronunciar o fonema; na coluna (b), é ilustrada a distribuição de energia ao longo do tempo; e na coluna (c), estão os valores de intensidade de energia distribuídos ao longo da frequência.

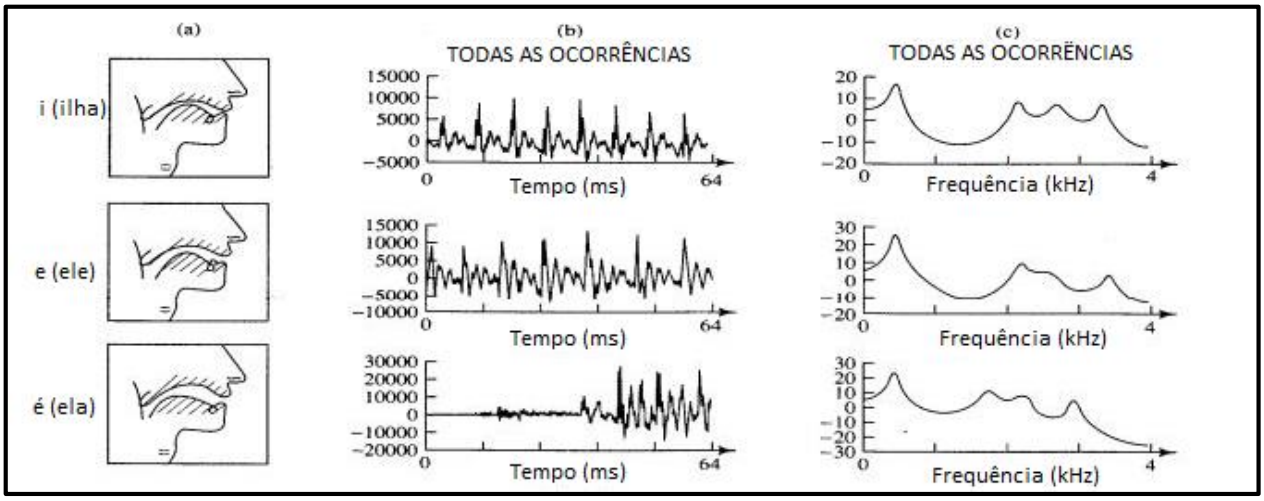


Figura 13 - Exemplo de sinais de alguns fonemas
Fonte: Adaptado de http://cs.haifa.ac.il/~nimrod/Compression/Speech/S1Basics2010.pdf

2.3.2 Modelos Ocultos de Markov (MOM)

Os Modelos Ocultos de Markov (MOM) são sistemas estocásticos finitos que tem como finalidade modelar um processo oculto de Markov. Este processo se caracteriza por não ter memória inerente, ou seja, cada estado do processo num instante de tempo T_i depende somente do estado em T_{i-1}; os estados anteriores a T_{i-1} em nada interferem no estado T_i. Mais adiante, um processo oculto de Markov possui a característica de ter estados não observáveis, como o processo de retirar objetos de cores diferentes de dentro de duas urnas escondidas; o resultado final (cor do objeto retirado) pode ser observado, mas o estado intermediário (de qual urna veio o objeto) está oculto para um observador (COSTA, 2008).

A Figura 14 mostra um exemplo de transição entre estados em um MOM. Os estados X1, X2 e X3 não são observáveis, enquanto que y1, y2, y3 e y4 o são. Ainda assim, as transições entre os estados não observáveis são previsíveis e podem ser estudadas estatisticamente.

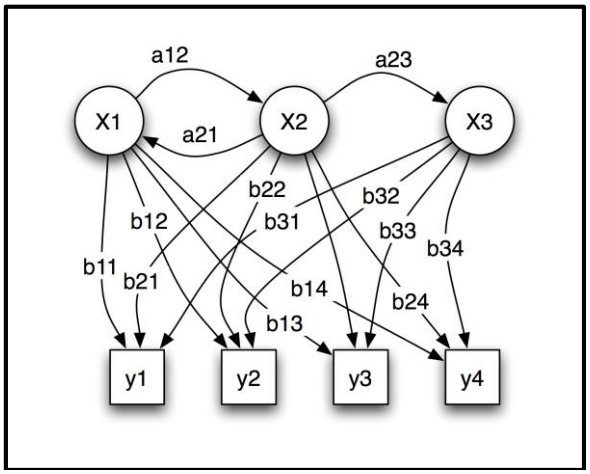


Figura 14 - Transição entre estados em um MOM
Fonte:
http://upload.wikimedia.org/wikipedia/commons/2/2e/HiddenMarkovModel.png

No contexto do reconhecimento de fala, Rabiner (1989) afirma que o processo de formação de uma palavra, definido como uma sequência de fonemas (estados) e transições entre eles, pode ser considerado como um processo oculto de Markov. Isto porque, apesar de ser possível observar alguns estados (som dos fonemas) existem estados ocultos, como repetição contínua de um determinado fonema e períodos curtos de silêncio. Assim os MOM são excelentes modelos para representar a fala humana.

2.3.3 Modelagem de sinais da fala

Para que seja possível trabalhar com os sinais acústicos gerados pela fala, é preciso utilizar modelos que representem da melhor maneira possível aquilo que foi dito pelo usuário. Este processo se baseia em uma análise estatística complexa que permite a inferência da palavra ou frase pronunciada (DYMARSKI, 2011).

Três componentes fundamentais desta análise estatística – Modelo Acústico (MA), Modelo de Linguagem (ML) e Dicionário Fonético – serão abordados a seguir, com o intuito de mostrar como um reconhecedor de fala decide qual é a palavra ou frase mais provável, após uma entrada de sinal de fala.

Grande parte dos reconhecedores de fala se baseia nos Modelos Ocultos de Markov (MOM) para construir os modelos acústicos e de linguagem. No caso do MA, os MOM são utilizados para converter o sinal de voz digitalizado em uma distribuição de parâmetros que maximiza a probabilidade de a sequência de fonemas pronunciada se assemelhar a alguma palavra previamente gravada no MA. Já para o ML, os MOM buscam maximizar a probabilidade de uma sequência de palavras pronunciada se assemelhar a uma sentença previamente gravada no ML (SILVA *et al.*, 2004).

2.3.3.1 Modelo Acústico (MA)

O Modelo Acústico é o componente do sistema de reconhecimento de fala responsável por definir, dada uma entrada de áudio, a sequência de fonemas mais provável, através da análise dos parâmetros calculados com os MOM.

Para que o reconhecedor seja confiável, o MA deve ser gravado no idioma que se quer reconhecer. Fazendo isto, o MA terá as informações precisas da pronúncia de cada fonema naquele determinado idioma, salvo diferenças de dialetos e locutores não-nativos. Além disso, o MA deve possuir a maior quantidade possível de áudio gravado por pessoas de diferentes timbres de voz, para que o reconhecedor possa funcionar com qualquer locutor.

Para o presente trabalho, o MA utilizado foi o LaPSAM, do grupo de pesquisa Fala Brasil do Laboratório de Processamento de Sinais da Universidade Federal do Pará, em sua versão mais recente, v1.5 de 2012. Este Modelo foi gravado utilizando uma taxa de amostragem de 20,05 kHz e possui um total de 15h e 41min de áudio.

2.3.3.2 Modelo de Linguagem (ML)

Analogamente ao MA, o ML também define um caminho de maior probabilidade de correspondência com o modelo, mas em relação à conexão das palavras dentro de uma sentença. O caminho buscado leva em consideração regras

de gramática (ordem lógica das classes gramaticais e concordância) e de semântica (significado mais lógico).

Devido ao fato de o presente trabalho buscar um reconhecedor de palavras isoladas, o ML tem por função somente conectar o período de silêncio inicial, a palavra pronunciada e o período de silêncio final.

Foi utilizado para esta aplicação o LaPSLM, também do grupo de pesquisa Fala Brasil. O modelo possui um total de 1,6 milhão de frases e 64.972 palavras. A última atualização do modelo foi feita em 2012.

2.3.3.3 Dicionário Fonético

O Dicionário Fonético define o vocabulário que será reconhecido após o MA converter a entrada de áudio em uma sequência fonética, para que o sistema decida qual palavra será mostrada na saída do reconhecedor. Ele pode ser editado pelo usuário, inclusive para adicionar palavras que não tenham sido gravadas no MA. Para isso, basta o usuário saber qual a sequência fonética da palavra e escrevê-la de maneira correta. É possível ainda escrever a mesma palavra de saída mais de uma vez, para que o sistema reconheça diferentes pronúncias de uma mesma palavra.

O Dicionário Fonético utilizado foi o UFPAdic 3.0 de 2012, também do grupo de pesquisa Fala Brasil. Devido à limitação de capacidade de processamento do presente trabalho, o dicionário foi manualmente reduzido para incluir somente as palavras-chaves que serão utilizadas na aplicação. A versão final do dicionário utilizada pode ser encontrada no APÊNDICE A.

2.4 PROGRAMAS DE RECONHECIMENTO DE FALA EXISTENTES

Existem diversos reconhecedores de fala disponíveis no mercado em praticamente todos os idiomas. Entretanto, estes são demasiadamente caros e de código fechado, o que dificulta a adaptação do programa a uma aplicação específica.

Por este motivo, para o presente trabalho foi selecionado o decodificador *Julius*, o qual, além de ser gratuito, possui código aberto.

2.4.1 *Julius*

O *Julius* é um programa decodificador com código aberto usado para reconhecimento de fala contínua. Desenvolvido sob os sistemas operacionais *Windows* e *Linux*, possui um grande vocabulário, alto desempenho e versatilidade. Com ele, pode-se executar o reconhecimento de voz tanto em tempo real através de dispositivos de entrada como o microfone quanto de arquivos de áudio previamente armazenados nos formatos WAV (*Waveform Audio File* ou arquivo de áudio de forma de onda) ou RAW (do inglês, cru; arquivos que possuem as informações exatamente como foram captadas pelo microfone). A primeira versão foi lançada em 1996 no Japão e, até hoje, está em constante evolução devido às necessidades de aplicação e aos desafios técnicos (LEE, 2010).

Para o correto funcionamento do *Julius*, existem modelos que definem as propriedades linguísticas do idioma escolhido e também determinam a precisão do reconhecimento, são eles: Dicionário, Modelo de Linguagem e Modelo Acústico.

Os modelos acústicos e de linguagem são adaptáveis, por isso é possível construir vários tipos de reconhecedores de fala de acordo com a aplicação desejada. Além disso, o *Julius* é uma ferramenta independente de idioma, ou seja, ele pode ser utilizado para reconhecer qualquer idioma desde que sejam fornecidos o dicionário, Modelo de Linguagem e Modelo Acústico.

O funcionamento do *Julius* é bastante simples: ao ser executado, ele busca a entrada de áudio indicada e o arquivo de configuração com extensão .jconf, onde são apontados quais devem ser os modelos acústicos e de linguagem a serem utilizados. Além disso, o Dicionário Fonético também é carregado pelo programa.

Caso o carregamento desses arquivos ocorra sem falhas, o *Julius* entra em um laço de repetição, no qual cada ciclo termina com o fim da entrada de áudio do usuário. Assim, para cada comando dado, o programa disponibiliza algumas saídas, relacionadas ao resultado da decodificação. A qualidade destas saídas depende da

capacidade de processamento do sistema utilizado, do refinamento dos modelos acústicos e de linguagem e do equipamento de entrada de áudio.

Dentre as saídas disponibilizadas pelo *Julius* destacam-se: *sentence1*, *cmscore1* e *score1*. Elas significam respectivamente: a palavra ou frase de maior probabilidade calculada, nível de confiança da palavra ou frase disponibilizada e coeficiente de Viterbi do algoritmo de cálculo de sequência de fonemas.

Em outras palavras, dada uma entrada de áudio, uma análise estatística é realizada para verificar, dentro das opções do Dicionário Fonético, qual sequência de fonemas será disponibilizada para o usuário. Esta análise produz diferentes resultados, cada um com uma probabilidade específica; o que será gravado na variável *sentence1*, será aquele de maior probabilidade. Esta probabilidade será gravada na variável *cmscore1*, nível de confiança do cálculo realizado. Finalmente, o coeficiente de Viterbi define o comprimento do caminho entre fonemas traçado pelo algoritmo de cálculo; este valor é gravado na variável *score1*.

O *Julius* está disponível no seguinte endereço eletrônico: http://julius.sourceforge.jp. Nele, pode-se encontrar a última versão do programa, vários modelos acústicos e de linguagem, documentos e referências de códigos fontes.

2.4.2 Projeto Coruja

O grupo de pesquisa FalaBrasil do Laboratório de Processamento de Sinais (LaPS) da Universidade Federal do Pará (UFPA), com apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), visa elaborar e disponibilizar recursos para sistemas de reconhecimento automático de voz para o Português Brasileiro. No endereço eletrônico do grupo é possível encontrar dicionários fonéticos, modelos de linguagem e modelos acústicos, que podem ser utilizados nas mais diversas aplicações (LABORATÓRIO DE PROCESSAMENTO..., 2013). Um exemplo de programa de reconhecimento de voz em Português Brasileiro desenvolvido por este grupo é o Coruja.

O Coruja é uma API (*Application Programming Interface* ou Interface de Programação de Aplicativos) desenvolvida na linguagem de programação C++ que

possibilita o controle em tempo real do decodificador de reconhecimento de voz *Julius* e de um Modelo Acústico.

Segundo Somera (2006) a interface de programação de aplicativos é um conjunto de rotinas, protocolos, ferramentas e padrões estabelecidos por um *software,* cujas funcionalidades são utilizadas por aplicações. Resumidamente, a API permite que aplicações, as quais não desejam se envolver em detalhes de implementação (linguagem de baixo nível de programação), possam apenas utilizar os serviços do *software.* A Figura 15 mostra como a aplicação interage com o reconhecedor *Julius (software)* através da API. O *Julius* recebe informações do Modelo Acústico, do Modelo de Linguagem e de áudio e, através da API (Coruja), relaciona-se com a aplicação desejada.

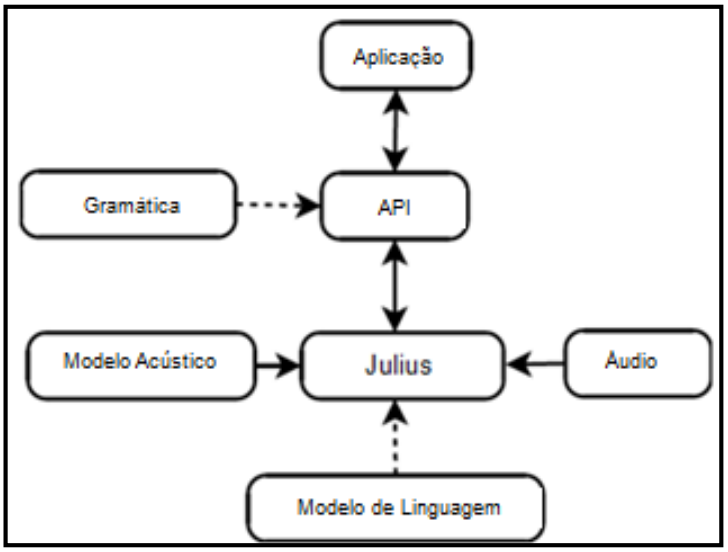


Figura 15 - API e aplicativos
Fonte: Adaptado de KLAUTAU et al. (2010).

A API Coruja suporta objetos compatíveis com o modelo de automação COM (*Component Object Model* ou Modelo de Objeto Componente), o qual possibilita acesso, manipulação, ajuste de propriedades e invocação de métodos de objetos de automação compartilhados que são utilizados por outras aplicações (KLAUTAU et al., 2010). A classe principal desta API é a *SREngine*, cujos métodos e eventos são mostrados na Tabela 3.

Tabela 3 - Principais métodos e eventos da API Coruja

Métodos/Eventos	Descrição Básica
SREngine	Método para carregar e inicializar o reconhecedor
loadGrammar	Método para carregar gramática SAPI XML
addGrammar	Método para carregar gramática nativa do *Julius*
startRecognition	Método para iniciar o reconhecimento
stopRecognition	Método para pausar/parar o reconhecimento
OnSpeechReady	Evento chamado quando o reconhecimento é ativado
OnRecognition	Evento chamado quando alguma sentença é reconhecida

Fonte: KLAUTAU et al. (2010).

A classe *SREngine* possibilita que o Coruja controle vários aspectos do decodificador *Julius* como: carregar os modelos acústico e de linguagem, habilitar e desabilitar o reconhecimento de voz e receber eventos e resultados vindos do *engine* de reconhecimento.

Uma gramática indica o método de reconhecimento que será empregado, por exemplo, informa quais palavras poderão ser reconhecidas e define as regras gramaticais. Para carregar e ativar uma gramática, o método *loadGrammar* é utilizado. Este método possui um conversor gramatical que permite a conversão automática de uma gramática de reconhecimento para o formato suportado pelo *Julius* (modelo de linguagem ou gramática livre de contexto). Para adicionar uma gramática a aplicação, utiliza-se o método *addGrammar* (LABORATÓRIO DE PROCESSAMENTO..., 2013).

Para iniciar o reconhecimento, ativar a gramática e habilitar o fluxo de áudio, o método *startRecognition* é invocado. Já para parar o reconhecimento, desativar a gramática e desabilitar o fluxo de áudio, o método *stopRecognition* é usado. Outros dois eventos foram criados para auxiliar o funcionamento da API*: OnSpeechReady* e *On*Recognition. O primeiro indica que o *engine* está ativado para reconhecimento e surge sempre que o método *startRecognition* é chamado. O segundo aparece toda vez que o resultado e o nível de confiança do reconhecimento estão disponíveis. Estes parâmetros são enviados para a aplicação através da classe *RecoResult* (KLAUTAU et al., 2010).

3 DESENVOLVIMENTO DO PROGRAMA DE RECONHECIMENTO DE VOZ

Neste capítulo serão abordadas as etapas de desenvolvimento do programa de reconhecimento de voz: desde a inicialização do minicomputador *Raspberry Pi* até o teste de validação do programa desenvolvido.

3.1 INICIANDO O R*ASPBERRY PI*

Com intuito de iniciar a implementação do trabalho em questão, foi realizada a instalação do sistema operacional no minicomputador *Raspberry Pi* e também foram feitos ajustes necessários na sua configuração inicial.

A Figura 16 mostra como foram conectados os seguintes periféricos no *Raspberry Pi* modelo B:

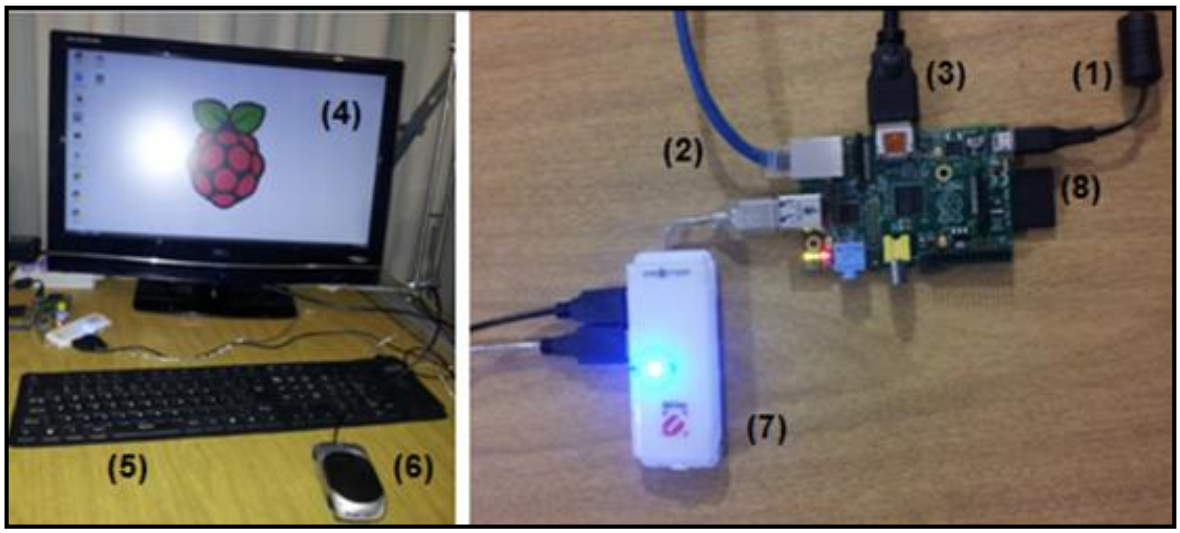


Figura 16 - Implementação *Raspberry Pi* + Periféricos
(1) Fonte de alimentação; (2) Cabo *Ethernet;* (3) Cabo HDMI; (4) Monitor; (5) Teclado; (6) Mouse; (7) HUB; (8) Cartão SD.
Fonte: Autoria própria.

A fonte de alimentação é conectada à entrada micro USB do *Raspberry*

Pi. Para o bom funcionamento do computador, recomenda-se a utilização de uma fonte que forneça pelo menos 5V e 700mA. Se a fonte de alimentação não for adequada, o funcionamento do *Raspberry Pi* pode ser afetado (*RASPBERRY PI...,* 2012). A fonte de alimentação utilizada é de 5V com 2A, para garantir não somente a energização adequada do processador, como também dos periféricos utilizados: mouse e teclado.

O cabo *Ethernet* utilizado para conexão com a rede local. A porta *Ethernet* do RPi pode ser conectada diretamente a um roteador ou diretamente em outro computador conectado a Internet, sem a necessidade da ligação *cross-over* (*RASPBERRY PI...,* 2012) . O cabo HDMI (*High-Definition Multimidia Interface* ou Interface Multimídia de Alta Definição) é usado para conectar o *Raspberry Pi* ao monitor LCD (*Liquid Crystal Display* ou Display de Cristal Líquido).

O RPi possui apenas duas entradas USB. Para que mais dispositivos possam ser conectados ao minicomputador, é necessária a utilização de um HUB (concentrador). O teclado e o mouse foram conectados nas entradas USB disponíveis.

O *Raspberry Pi* não possui memória de armazenamento interna, nem sistema operacional incluso, por isso um cartão SD (*Secure Digital*) deve ser utilizado. Pode-se comprar um cartão SD com o sistema operacional pré-instalado ou pode-se utilizar um cartão SD com alta capacidade (*HC – High Capacity*) e, pelo menos, 4 GB de memória para instalar o sistema operacional. A equipe optou pela segunda opção e, em um cartão SDHC de 8 GB classe 4[7], instalou o sistema operacional *Raspbian* seguindo os passos recomendados pelo manual do fabricante do RPi (*RASPBERRY PI*..., 2013):

1. Colocar o cartão SD de 4 GB ou mais no computador.
2. Formatar o cartão SD (computador com Windows):
    a. Baixar a Ferramenta de Formatação da Associação SD acessando o endereço na Internet*:* https://www.sdcard.org/downloads/formatter_4/eula_windows/
    b. Instalar e executar a Ferramenta de Formatação no computador.

[7] A Classe de um cartão de memória está relacionada com a velocidade de transferência de dados. Um cartão SD classe 4 é capaz de transferir a uma velocidade de, pelo menos, 4MB/s.

c. Configurar a opção "*FORMAT SIZE ADJUSTMENT*" para "*ON*" no menu "Opções".

d. Verificar se o cartão SD inserido corresponde ao selecionado pela Ferramenta de Formatação.

e. Clicar no botão “*Format*”.

A Figura 17 serve para facilitar o entendimento das etapas de “b” até “e”, pois mostra a tela que aparece da Ferramenta de Formatação.

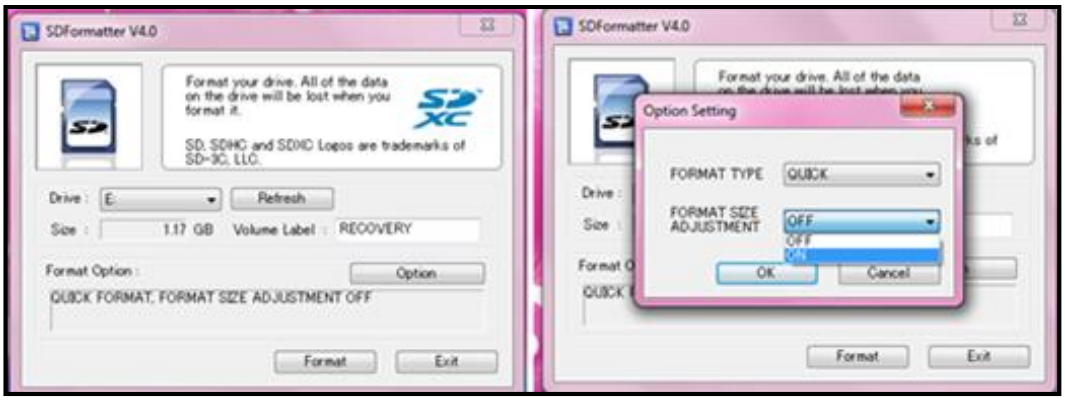
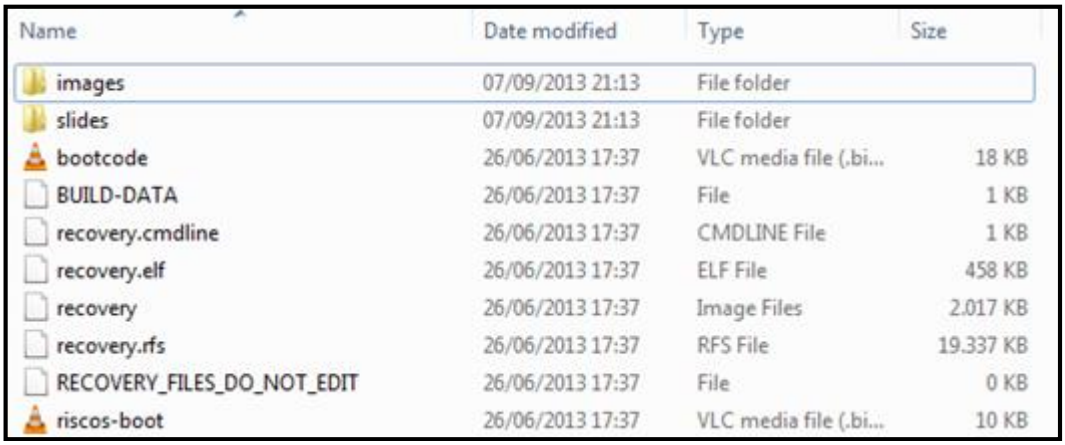


Figura 17 - Formatando cartão SD
Fonte: Autoria própria.

3. Baixar o *New Out Of Box Software* (NOOBS) no endereço na Internet: downloads.raspberrypi.org/noobs

4. Descompactar o arquivo baixado.

5. Copiar os arquivos descompactados, os quais estão representados na Figura 18, no cartão SD que foi formatado.



Name	Date modified	Type	Size
images	07/09/2013 21:13	File folder	
slides	07/09/2013 21:13	File folder	
bootcode	26/06/2013 17:37	VLC media file (.bi...	18 KB
BUILD-DATA	26/06/2013 17:37	File	1 KB
recovery.cmdline	26/06/2013 17:37	CMDLINE File	1 KB
recovery.elf	26/06/2013 17:37	ELF File	458 KB
recovery	26/06/2013 17:37	Image Files	2.017 KB
recovery.rfs	26/06/2013 17:37	RFS File	19.337 KB
RECOVERY_FILES_DO_NOT_EDIT	26/06/2013 17:37	File	0 KB
riscos-boot	26/06/2013 17:37	VLC media file (.bi...	10 KB

Figura 18 - Arquivos NOOBS
Fonte: Autoria própria.

6. Colocar o cartão SD no RPi.
7. Conectar o HUB no USB do RPi e ligar o mouse e o teclado nas entradas USB do HUB.
8. Conectar o cabo HDMI, ligando o RPi ao monitor.
9. Conectar o cabo *Ethernet* ao roteador ou outro computador com acesso a Internet.
10. Conectar a fonte de alimentação na entrada micro USB do RPi.
11. Com o monitor ligado, ligar a fonte de alimentação.

Após realizados estes passos, o *Raspberry Pi* inicializa-se na tela NOOBS, como mostra a Figura 19, e deve-se escolher um dos sistemas operacionais para ser instalado.

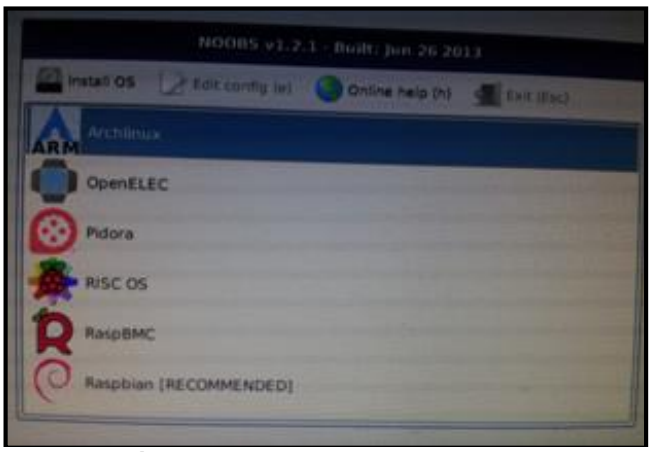


Figura 19 - Tela NOOBS
Fonte: Autoria própria.

O *Raspbian*, que é sistema operacional baseado no *Debian* recomendado pela Fundação *Raspberry Pi*, foi escolhido e instalado conforme Figura 20.

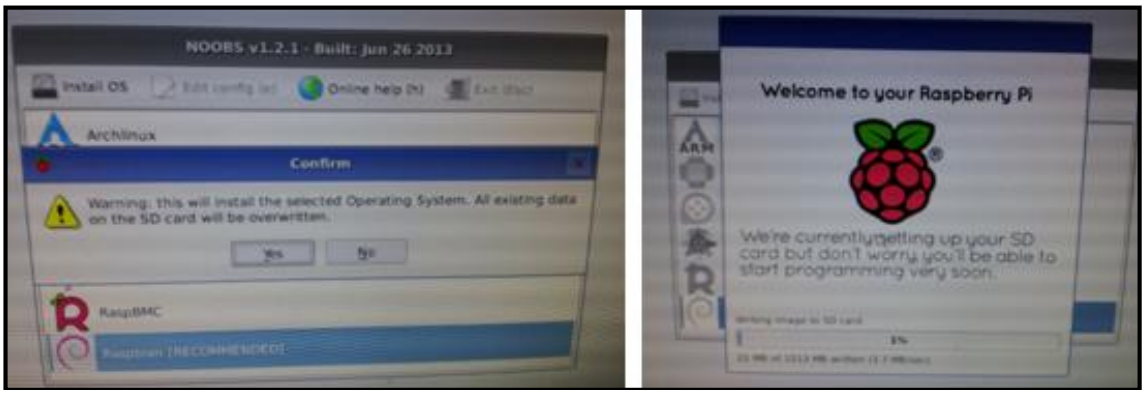


Figura 20 - Instalação *Raspbian*
Fonte: Autoria própria.

Quando a instalação do *Raspbian* tiver sido finalizada, é necessário realizar a configuração básica do *Raspberry Pi.* Para isso, foram utilizadas as seguintes opções de ajuste:

1. *Expand Filesystem* (Expandir Arquivos do Sistema): para permitir que toda memória de armazenamento do cartão SD esteja disponível para o sistema operacional, é necessário executar esta opção.
2. *Enable Boot to Desktop* (Habilitar Inicialização para a Área de Trabalho): permite escolher o modo de inicialização do RPi (Área de Trabalho ou Linhas de Comando).
3. *Internationalisation Options* (Opções de Internacionalização): permite configurar a localização e também o fuso horário. A Figura 21 mostra a configuração feita para alterar o local de uso do RPi, do Reino Unido para o Brasil.

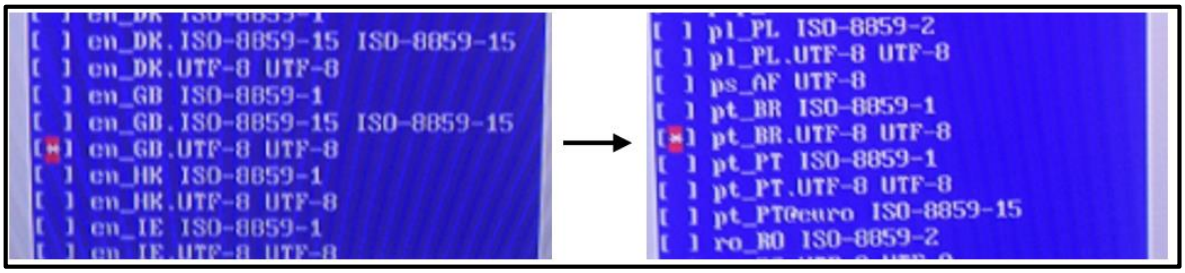


Figura 21 - Configuração da localização do RPi
Fonte: Autoria própria.

A tela de configuração com as todas opções de ajuste está representada na Figura 22.

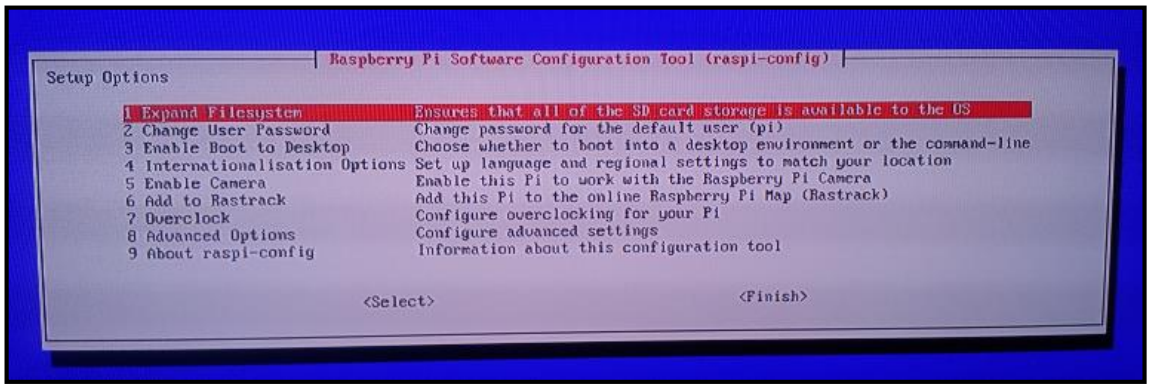


Figura 22 - Tela de configuração do *Raspberry Pi*
Fonte: Autoria própria.

Após realizada está primeira configuração. O *Raspberry Pi* está pronto para uso.

3.2 INSTALANDO O *JULIUS* E O MODELO ACÚSTICO DO CORUJA

Após muita pesquisa e leitura, decidiu-se utilizar o programa *Julius* combinando-o com o modelo acústico do projeto Coruja (LaPS) para realizar o reconhecimento de voz necessário para o acionamento da plataforma elevatória. Este decodificador foi escolhido por ser gratuito, possuir código aberto e compatível com o sistema operacional do RPi.A instalação foi realizada utilizando o sistema operacional

Raspbian (32-bit), o qual corresponde a uma versão do *Debian* otimizada para o microcomputador *Raspberry Pi.*

Todos os passos foram realizados com o super usuário *root* logado e com conexão a Internet. Foram utilizados o tutorial para instalação do *Julius* usando o *Raspberry Pi*[8] e o modelo acústico disponibilizado no grupo *Google:* Coruja – Reconhecimento de Voz em PB[9] *.*

Primeiramente, o *firmware* do *Raspberry Pi* foi atualizado com os comandos abaixo:

```
apt-get install update
apt-get install upgrade
```

Alguns pacotes e bibliotecas precisaram ser instalados para que o sistema funcionasse corretamente.

```
apt-get install alsa-tools alsa-oss flex zlib1g-dev libc-
bin libc-dev-bin python-pexpect libasound2 libasound2-dev
cvs build-essential libesd0-dev libsndfile1-dev
```

Uma vez que o dispositivo interno de som do *Raspberry Pi* é definido como plughw:0,0; quando o microfone USB é anexado, este é atribuído ao plughw:1,0. Para verificar se o microfone USB estava funcionando, dez segundos de áudio foram gravados utilizando o comando:

```
arecord -d 10 -D plughw:1,0 test.wav [10]
```

Em seguida o arquivo de áudio foi reproduzido:

```
aplay test.wav
```

Com o áudio funcionando, a versão CVS do *Julius* foi carregada:

---

[8] *Speech Recognition Using The Raspberry Pi*: http://www.aonsquared.co.uk/raspi_voice_control

[9] Tutorial de instalação do *Julius* + modelos acústicos do Coruja no Ubuntu x86: https://groups.google.com/forum/?hl=pt-BR#!topic/coruja-users/Sjl8InZ11B8

[10] A opção –d 10 define a duração (10 segundos) da gravação e a opção –D plughw:1,0 atribui o dispositivo que será usado para gravação (microfone USB).

```
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.jp:/
cvsroot/ julius co julius4
```

Algumas opções de compilação foram configuradas pelas variáveis de ambiente:

```
export CFLAGS="-O2 -mcpu=arm1176jzf-s -mfpu=vfp -mfloat-
abi=hard -pipe -fomit-frame-pointer"
```

Para instalar o *Julius*, entrou-se na pasta recém-criada julius4 (na pasta *root*) e os comandos foram efetuados:

```
cd julius4
./configure --with-mictype=alsa
make
make install
```

Para garantir que o programa foi instalado corretamente, foi verificada a existência do arquivo:

```
Julius –version
```

O modelo acústico a ser utilizado foi o LaPSAM 1.5, o qual foi desenvolvido pelos pesquisadores da UFPA no âmbito do projeto Coruja[11].

Após o carregamento do modelo acústico LaPSAM através do endereço eletrônico, os arquivos foram extraídos na pasta *root.* Para descompactar o modelo acústico com extensão .rar, é necessário instalar o pacote *unrar* no *Raspberry Pi*:

```
apt-get install unrar-free
wget http://www.rarlab.com/rar/unrarsrc-4.2.4.tar.gz
gunzip -v unrarsrc-4.2.4.tar.gz
tar -xvf unrarsrc-4.2.4.tar

cd unrar/
make -f makefile.unix
```

[11] Endereço para obter o modelo acústico LaPSAM http://www.laps.ufpa.br/falabrasil/files/jlapsapi/for_coruja1.5/coruja_jlapsapi1.5.rar

```
cp unrar /usr/local/bin

update-alternatives --install /usr/bin/unrar unrar
/usr/local/bin/unrar 10
update-alternatives --display unrar
update-alternatives --config unrar
```

Após extração, os seguintes arquivos estavam disponíveis:

```
root/coruja_jlapsapi
├── dic.temp
├── julius.jconf
├── LaPSAM1.5.am.bin
├── LaPSAM1.5.tiedlist
└── LaPSLM1.5.lm.bin
```

Editou-se o dicionário (dic.temp) somente com o vocabulário necessário para o acionamento da plataforma (APÊNDICE A). O arquivo julius.jconf também foi configurado de acordo com a aplicação do programa (ANEXO A). Foram retirados os comentários das linhas de *realtime* e *–input mic* e comentadas as linhas *norealtime* e *–input rawfile.*

```
...
-realtime
#-norealtime
...
#-input rawfile
-input mic
```

No terminal, entrou-se na pasta em que está guardado o arquivo julius.jconf, definiu-se o dispositivo de áudio a ser utilizado para o reconhecimento de voz e executou-se o programa através dos comandos:

```
cd root/coruja_jlapsapi
export ALSADEV="plughw:1,0"
julius -input mic -C julius.jconf
```

Vale frisar que o usuário deve sempre digitar os textos e códigos, pois, quando se usa as funções de copiar e colar, o sistema do *Raspberry Pi* não reconhece alguns caracteres gerando erros na compilação.

3.3 ESCREVENDO O CÓDIGO *PYTHON*

O código do programa foi escrito em *Python*, linguagem compatível com o sistema operacional *Raspbian* e com diversas fontes de informações para esclarecimento de dúvidas.

A função básica do programa desenvolvido é unificar a saída do *software Julius* com a lógica de acionamento da plataforma. Com a intenção de não modificar o funcionamento do sistema como está instalado atualmente, esta lógica foi elaborada de tal forma a operar em paralelo com o acionamento manual (botoeira) já existente. Com isso, o programa teve limitações diretamente relacionadas à inflexibilidade para modificações do acionamento mecânico.

A lógica é baseada em interpretar o comando de voz dado pelo usuário, decodificando-o através do *Julius* e, de acordo com uma lista de palavras pré-definidas (dicionário fonético), acionar ou não as saídas do sistema utilizando relés. Uma interface com o usuário, constituída por elementos visuais (LED*s*), também é acionada para indicar o estado do sistema: sistema de reconhecimento de voz habilitado, plataforma acionada ou plataforma parada.

A seguir, tem-se trechos do código desenvolvido referentes à parte de configuração inicial do programa: inclusão de bibliotecas e declaração de variáveis globais com respectivos valores de referência. O código na integra está disponível no APÊNDICE B.

```
#Constantes de nivel de confianca e coeficiente de Viterbi
NivelConf = 0.4
CoefViterbi = -10000

#Biblioteca de execucao automatica de aplicativos
import pexpect

#Biblioteca de busca de correspondencia entre strings
import re
```

Uma grande vantagem do programa desenvolvido é a facilidade em alterar os valores das variáveis globais como: NivelConf e CoefViterbi, que representam, respectivamente o nível de confiança e o coeficiente de Viterbi. Estas variáveis são

parâmetros para o reconhecimento de voz e definem o nível de aceitação do programa em relação à entrada de voz[12].

3.4 TESTES DE VALIDAÇÃO

Um teste de confiabilidade foi realizado a fim de validar o programa de reconhecimento de voz *Julius* escolhido como base do sistema. A intenção deste teste era de identificar as palavras que o programa reconhece com maior facilidade para então definir os comandos que seriam utilizados para acionamento da plataforma elevatória.

Selecionaram-se as palavras que poderiam ser usadas como comandos (ativa, ative, ativar, aciona, acione, acionar, liga, ligue, ligar, anda, ande, andar, vai, para, pare, parar, parada, parou, socorro) e as definiram como vocabulário fonético do *Julius*, o qual foi executado através dos comandos:

```
cd root/coruja_jlapsapi
export ALSADEV="plughw:1,0"
julius -input mic -C julius.jconf
```

Com a finalidade de coletar dados representativos para o estudo, um total de quinze pessoas, com idade, gênero e origem variados, participou do teste. Cinco amostras de cada palavra foram coletadas para cada pessoa, registrando os parâmetros de nível de confiança e coeficiente de Viterbi. Com estas informações foi calculada a média dos resultados obtidos. Separam-se as palavras em dois grupos para análise: palavras de acionamento e palavras de parada da plataforma. A Figura 23 mostra o gráfico resultante do primeiro grupo.

[12] Para maiores detalhes sobre estas variáveis, consultar 2.4.1 Julius.

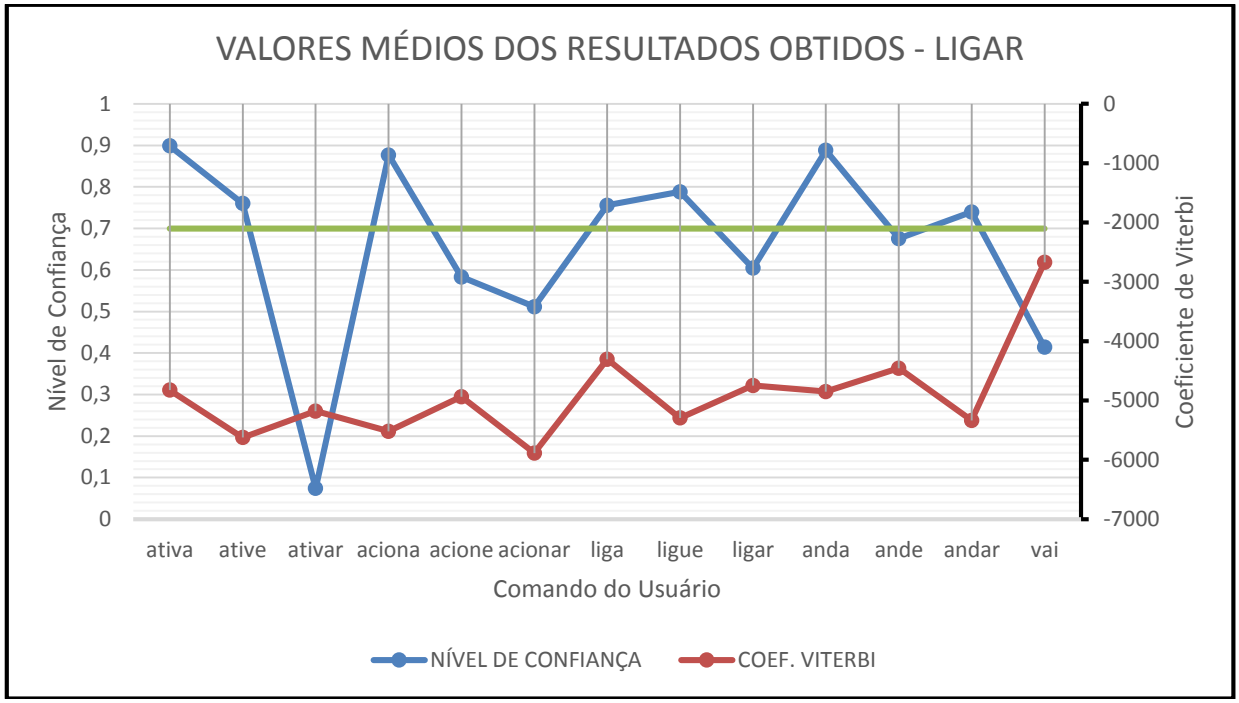


Figura 23 - Valores médios dos resultados obtidos – Acionamento
Fonte: Autoria própria.

Analisando o gráfico obtido, percebe-se que algumas palavras tiveram valores médios de respostas melhores que outras. Com isso, foi possível selecionar as opções de palavras mais adequadas, sendo estas definidas como as respostas com nível de confiança maior que 70%.

Para a utilização do usuário final, foram selecionadas as seguintes palavras para o acionamento da plataforma:

- Ativa;
- Ative;
- Aciona;
- Liga;
- Ligue;
- Anda.

Salienta-se que, apesar de ter obtido uma resposta média maior que 70%, a palavra “Andar” foi descartada devido ao fato de ser frequentemente interpretada como “Parar”. Como estas duas palavras referem-se a comandos opostos, optou-se pela sua não utilização.

Analogamente, para o grupo de palavras de parada da plataforma, obteve-se o gráfico representado na Figura 24.

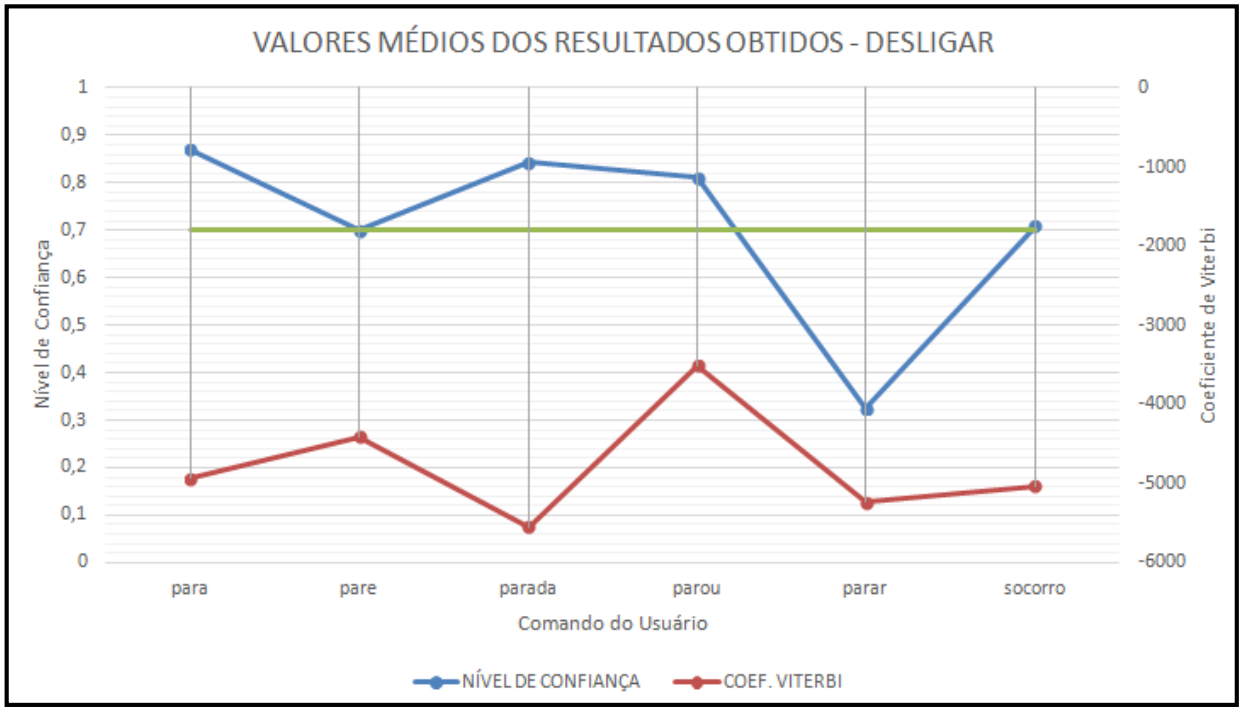


Figura 24 - Valores médios dos resultados obtidos – Parada
Fonte: Autoria própria.

Mais uma vez, foram selecionadas as palavras cujas respostas de nível de confiança foram maiores que 70%. Assim, para o comando de parada da plataforma, as palavras são:

- Para;
- Parada;
- Parou.

Para ambos os grupos de palavras, percebeu-se que as respostas de coeficiente de Viterbi foram bastante aceitáveis em praticamente todos os casos. Em outras palavras, em nenhum caso esta resposta média foi inferior ao valor estipulado como aceitável na lógica de programação (-10.000). Isto se explica pelo fato de o dicionário fonético utilizado possuir um vocabulário reduzido, ou seja, o reconhecedor tende a não confundir duas palavras diferentes.

4 INSTALAÇÃO E TESTES FINAIS DO SISTEMA

Este capítulo tratará da instalação do sistema de reconhecimento de voz para automatização da plataforma elevatória assim como dos testes realizados.

4.1 MATERIAIS E CUSTOS

Para desenvolver o sistema de reconhecimento de voz, além dos componentes listados na Tabela 4, foram utilizados os seguintes periféricos: monitor, teclado e mouse para inicialização do sistema. O custo total foi de R$ 316,51.

Tabela 4 - Custos para a construção do sistema de reconhecimento de voz proposto neste trabalho

Descrição	**Preço**
Fonte 5V 2A	R$ 19,00
Cabo HDMI	R$ 16,00
Cartão SD 8GB SDHC	R$ 27,68
Raspberry Pi	R$ 159,65
HUB	R$ 24,00
Placa com 4 relés	R$ 18,78
Sensor de presença PIR	R$ 19,80
Microfone de lapela com fio para câmera filmadora/ *notebook*/ pc	R$ 7,80
Jumper fio 40 peças de 20 cm Macho/Fêmea	R$ 11,28
Jumper fio 40 peças de 20 cm Fêmea/Fêmea	R$ 28,60
Componentes da placa de interface	R$ 14,00
TOTAL	R$ 316,51

Fonte: Autoria própria.

Desde que o programa de reconhecimento esteja em execução, os periféricos: monitor, teclado e mouse, podem ser desconectados do sistema. A Figura 25 exibe os componentes que permanecem após a inicialização do programa, são eles:

Figura 25 - Componentes do sistema de reconhecimento de voz
(1) Sensor PIR; (2) Microfone de eletreto; (3) HUB; (4) Fonte de alimentação; (5) Cartão SD; (6) *Raspberry Pi*; (7) Placa com reles; (8) Placa de interface; (9) *Jumpers*.
Fonte: Autoria própria.

4.2 INSTALAÇÃO DO SISTEMA NA PLATAFORMA ELEVATÓRIA

O PROTA disponibilizou uma plataforma elevatória para que o sistema de reconhecimento de voz fosse implantado. Esta plataforma foi testada e estudada, a fim de verificar quais seriam as intervenções necessárias para adicionar o comando de voz paralelo ao existente.

Em primeira análise, foi verificado que uma pessoa pode subir e pode acionar uma botoeira posicionada na lateral da plataforma com rampa. Esta botoeira, indicada na Figura 26, conta com dois botões: um verde para acionamento e parada, e outro vermelho para paradas de emergência, porém este não estava conectado.

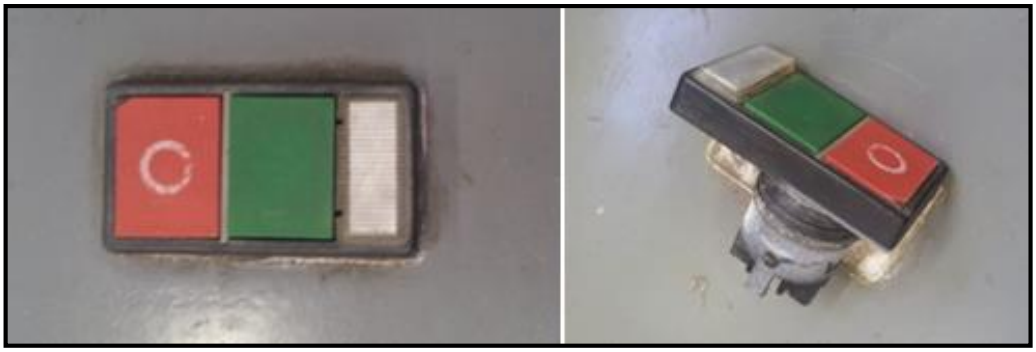


Figura 26 - Botoeira da plataforma elevatória
Fonte: Autoria própria.

Ainda na parte lateral, a remoção de uma tampa metálica possibilita o acesso ao motor e mecanismos da plataforma. A Figura 27 exibe as partes constituintes do sistema mecânico e de comando já existentes. É possível verificar que o motor utilizado faz parte de um sistema de portão eletrônico. Há uma placa eletrônica (1) para recepção de comandos através de controle remoto, paralelos aos comandos da botoeira. Existe também uma segunda placa (2) responsável pela operação do motor elétrico, com *bornes*, transformadores e componentes em geral.

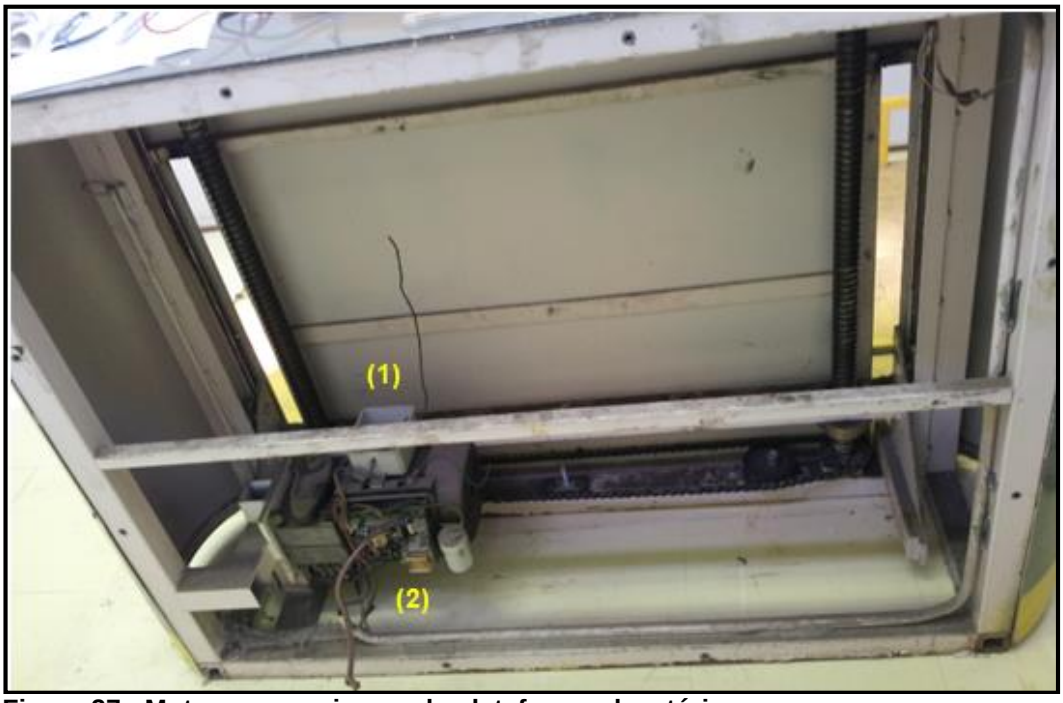


Figura 27 - Motor e mecanismos da plataforma elevatória
Fonte: Autoria própria.

A leitura de informações grafadas na placa, mostrada na Figura 28, permite identificar que o sistema é fornecido pelo fabricante PPA. Trata-se de um produto da linha de motores elétricos para portões, modelo Liger-20 (pivotante), Eurus-20 (deslizante).

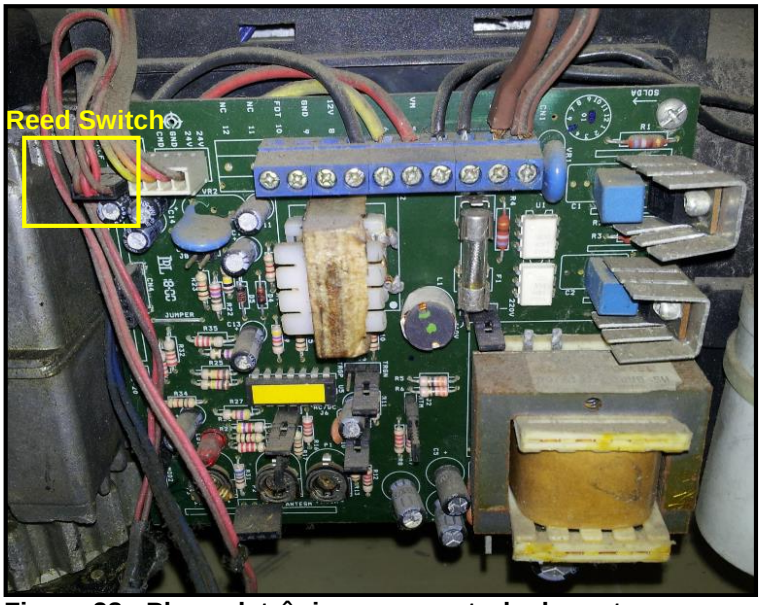


Figura 28 - Placa eletrônica para controle do motor
Fonte: Autoria própria.

Alguns testes simples foram realizados de forma a conhecer o funcionamento da plataforma e ações comandadas pela botoeira. A primeira experiência iniciou-se com a plataforma em seu ponto inicial, apoiada ao chão. Apertando o botão verde sobre o painel lateral, a plataforma sobe suavemente até o *reed switch* superior, responsável pela parada automática. Agora localizada no ponto superior, aperta-se novamente o botão verde e a descida, também suave, é habilitada até chegar ao ponto inferior onde está localizado outro *reed switch*.

O botão verde pode também ser pressionado com a plataforma em movimento. O acionar do botão durante sua descida faz com que a plataforma pare imediatamente e pressioná-lo novamente irá reverter o sentido do motor, elevando a plataforma. Caso a plataforma esteja subindo, o pressionar do botão realiza uma parada breve e em seguida, automaticamente, inicia-se o movimento de descida. Isto ocorre porque o sistema de portão eletrônico conta com uma função de antiesmagamento predefinida pelo fabricante, não sendo possível alteração. A Figura 29 ilustra o principio de funcionamento do sistema descrito acima.

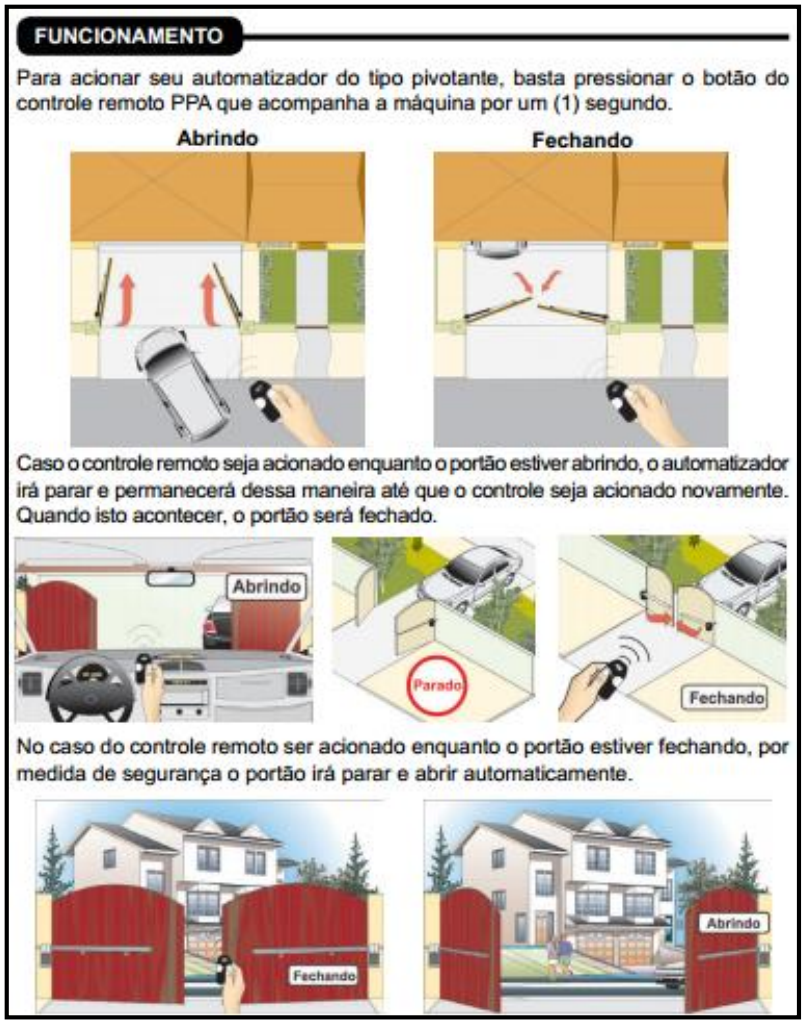


Figura 29 - Funcionamento do motor do sistema de portão eletrônico PPA
Fonte: PPA (2013).

Havendo uma possível queda de energia ou desconexão da rede elétrica a plataforma cessa o movimento, esteja subindo ou descendo. No retorno da energia ao sistema, a plataforma permanece parada até que o botão verde seja acionado, o que fará com que a plataforma inicie uma descida, retornando sempre à posição inicial.

Após a fase de análise, percebeu-se que a melhor solução para o acionamento da plataforma por comando de voz se daria após serem realizadas as seguintes alterações:

- O cabos do sensor *reed switch* foram derivados até a entrada do *Raspberry Pi*, para que fosse possível verificar se a plataforma está na posição mais baixa, mais alta ou em alguma intermediária;

- Os cabos da botoeira receberam uma ligação paralela advinda da saída do relé de acionamento;
- Os cabos de alimentação da plataforma foram conectados em série com a saída do relé de parada.

Assim, o funcionamento do sistema de reconhecimento de voz é o seguinte:

- Se a plataforma está situada no nível inferior e somente um comando de acionamento é dado: a plataforma subirá até o nível superior;
- Se a plataforma está situada no nível superior e somente um comando de acionamento é dado: a plataforma descerá até o nível inferior;
- Agora, se a plataforma está subindo ou descendo e um comando de parada é dado, a plataforma irá parar. Na sequência, se um comando de acionamento é falado: a plataforma descerá até o nível inferior ou até que outro comando de parada seja feito.

Para garantir a segurança do *Raspberry Pi*, foi construída uma placa de interface entre as GPIO e os componentes externos, com resistores e conectores que reduzem os efeitos negativos de surtos de tensão e mau-contatos. Nesta placa encontram-se também os LEDs de interface visual com o usuário – um LED amarelo indica que o sistema está pronto para receber o comando de voz, um verde indica que a plataforma está em movimento e um vermelho indica que está parada.

A fim de tornar o sistema mais robusto e proteger os componentes, estes foram alojados em uma caixa plástica para montagem de circuitos elétricos conforme mostrado na Figura 30.

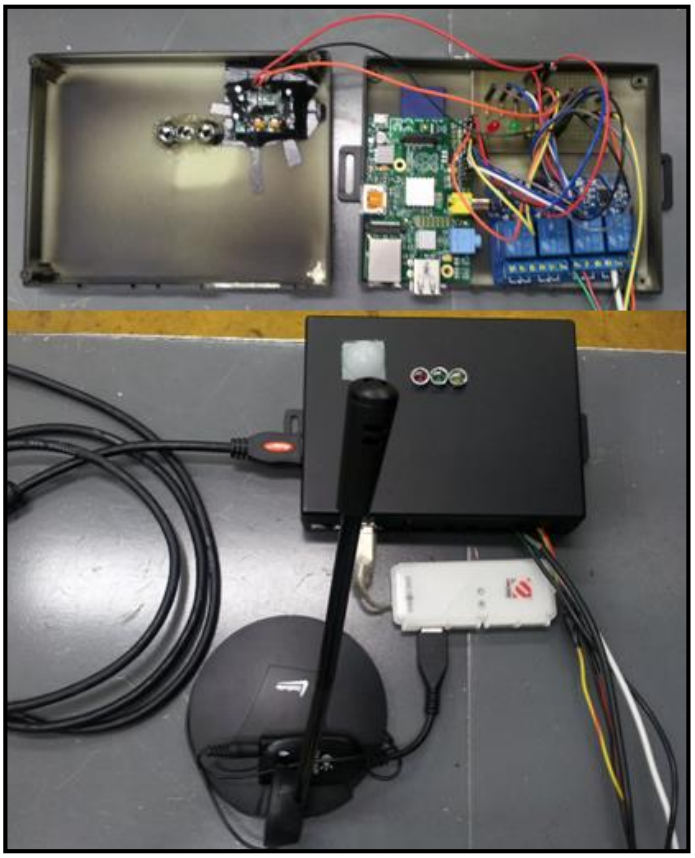


Figura 30 - Sistema de reconhecimento de voz na caixa plástica
Fonte: Autoria própria.

A Figura 31 mostra como ficou a implementação final do sistema de reconhecimento de voz na plataforma elevatória do PROTA.

Figura 31 - Implementação final do sistema de reconhecimento de voz
Fonte: Autoria própria.

O APÊNDICE C fornece as instruções de uso do sistema de reconhecimento de voz para automatização da plataforma elevatória.

4.3 TESTES FINAIS

Após a implementação do sistema, verificou-se que o tempo de resposta foi muito bom, ao se comparar com sistemas de reconhecimento de voz populares, como os de celulares e de sistemas operacionais, os quais possuem um tempo de resposta de aproximadamente dois segundos. Para o sistema desenvolvido, após o comando de voz, o tempo médio de acionamento ou parada da plataforma foi de três segundos.

Observou-se também que o microfone utilizado, apesar de fornecer excelentes resultados para o reconhecimento, possui limitações relacionadas a distância para aquisição do sinal de voz. A fim solucionar este problema, tentou-se utilizar um microfone de CFTV (Circuito Fechado de Televisão), que possui um alcance muito maior que o de eletreto – no caso do microfone adquirido, 150 m^2. No entanto, ao testá-lo, foi verificado que o reconhecimento foi prejudicado, porque o microfone, por ser muito sensível, captou muitos ruídos que eram interpretados como fonemas pelo programa. Como consequência, o coeficiente de Viterbi apresentou um valor extremamente baixo, ou seja, o algoritmo do decodificador percorreu uma trajetória entre fonemas muito grande para gerar uma resposta. Sendo assim, optou-se utilizar o microfone de eletreto, fixado em um suporte que o mantém a uma distância adequada para aquisição da entrada de voz por diversos usuários, por exemplo: crianças e cadeirantes.

Com todos os ajustes executados, o sistema foi testado por pessoas alheias ao desenvolvimento do projeto. O retorno foi positivo em todos os casos, sendo que a maior parte das falhas observadas foi devida às limitações do sistema mecânico, como lentidão no movimento, ruído e mau contato no sinal dos fins-de-curso (*reed switch*).

5 CONCLUSÃO

A automação relacionada a termos como acessibilidade, inclusão social e autonomia do usuário remete ao conceito de Tecnologia Assistiva. Esta, por meio da utilização de recursos e serviços específicos, visa a diminuir ou até mesmo neutralizar dificuldades funcionais ou de acesso encontradas por pessoas com necessidades especiais ou com mobilidade reduzida.

Nota-se que o recurso assistivo de comando de voz, escopo do presente trabalho, além de possibilitar o acionamento remoto de diversos sistemas, proporcionando conforto e comodidade, serve como um grande auxílio para execução de tarefas cotidianas. Quando utilizado por pessoas com necessidades especiais, este recurso aumenta a independência do usuário e melhora sua qualidade de vida.

Apesar do comando de voz ser uma tecnologia cujo mercado consumidor está em crescimento, poucos produtos são direcionados para a área da Tecnologia Assistiva. Existe uma dificuldade em se encontrar recursos para uso específico, como, por exemplo, recurso para acionamento de plataforma elevatória via comando de voz, que sejam financeiramente acessíveis.

Dessa maneira, após a realização da revisão bibliográfica, foi verificada a viabilidade de execução do trabalho com as características desejadas, as quais se resumem em: atender as pessoas com necessidades especiais, utilizar processamento via PC, ser de baixo custo, não depender de conexão *Ethernet* e ter uma resposta confiável.

Para a função de processamento de dados do sistema de voz para automatização de uma plataforma elevatória, utilizou-se o minicomputador de baixo custo *Raspberry Pi*. Este foi escolhido por possuir alta capacidade de processamento e ter interfaces de comunicação de fácil utilização. A programação do RPi é facilitada ao se explorar as funções disponibilizadas gratuitamente em diversos fóruns de discussões na Internet.

Analisaram-se os *softwares* de reconhecimento de fala já existentes, *Julius* e a API do projeto Coruja, e verificou-se a compatibilidade entre eles e a aplicação desejada através de teste, que evidenciou as palavras que seriam usadas como comandos para acionamento da plataforma (Ativa, Ative, Aciona, Liga, Ligue, Anda. Para, Parada e Parou).

O programa de reconhecimento de voz desenvolvido foi baseado nos modelos acústico e de linguagem do projeto Coruja e em um código em *Python* de autoria própria. A lógica de programação foi elaborada de modo a não interferir no atual funcionamento da plataforma elevatória (acionamento de portão eletrônico com comando manual), por isso o programa teve limitações diretamente relacionadas à inflexibilidade para modificações do acionamento mecânico.

Uma vez desenvolvido o código do programa, a placa de interface com o usuário foi montada e as ligações entre todos os componentes foram realizadas. Assim, o sistema foi implementado na plataforma elevatória.

O objetivo do trabalho foi atingido: desenvolveu-se um sistema eficiente, com bom tempo de resposta e de baixo custo de reconhecimento de voz para acionamento de uma plataforma elevatória. Além disso, o sistema é totalmente independente de conexão com a Internet ou qualquer outra fonte de dados. Este diferencial é bastante interessante, porque atualmente grande parte dos produtos existentes no mercado possui esta dependência.

A quem possa vir a interessar, algumas melhorias são sugeridas como forma de trabalhos futuros:

- Desenvolver uma interface gráfica com o usuário com mais recursos, por exemplo, com um monitor tipo RCA para mostrar, em tempo real, o estado do sistema através do terminal de comando do *Raspberry Pi*;
- Reformular a solução de automação, utilizando um controlador com maior capacidade de processamento, para reduzir ainda mais o tempo de resposta do sistema, pois no caso de vocabulário pequeno o processamento pode ser feito com maior desempenho;
- Gravar um novo Modelo Acústico, contendo somente as palavras que serão utilizadas para comando, também com a função de reduzir o tempo de resposta.

REFERÊNCIAS

ASCHER, David; LUTZ Mark. **Aprendendo *Python*:** Programação orientada a objetos. São Paulo: Bookman, Inc: 2004.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 9050:** Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. Rio de Janeiro, 2004.

BERSCH, Rita. **Introdução à tecnologia assistiva.** Centro Especializado em Desenvolvimento Infantil (CEDI), Porto Alegre, 2008.

BERSCH, Rita; **Sartoretto, Mara L. Assistiva: Tecnologia e Educação.** Brasil, 2013.Disponível em: <http://www.assistiva.com.br/>. Acesso em: 01 jul. 2013.

BRASIL. **Decreto 5.296/2004**, de 2 de dezembro de 2004. Regulamenta as Leis nºs 10.048, de 8 de novembro de 2000 e 10.098, de 19 de dezembro de 2000. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm>. Acesso em: 05 jun. 2013.

BOLTON, William. ***Mechatronics: A Multidisciplinary Approach.*** 4. ed. England: Pearson Education Limited, 2008.

BOLZANI, Caio Augustus M. **Residências Inteligentes**, 1. ed. São Paulo: Editora Livraria da Física, 2004.

CAMPOS, Augusto. **O que é Linux.** BR-Linux. Florianópolis, março de 2006. Disponível em <http://br-linux.org/faq-linux>. Acesso em 16 jul. 2013

CHAN, M.; ESTÈVE, D.; ESCRIBA, C.; CAMPO, E. ***A review of smart homes – Present state and fututre challenges.*** *Computer methods and programs in biomedicine*, n. 91, p. 55-81, 2008.

COMITÊ DE AJUDAS TÉCNICAS. **Ata da Reunião VII**, de dezembro de 2007, Comitê de Ajudas Técnicas, Secretaria Especial dos Direitos Humanos da Presidência da República (CORDE/SEDH/PR). Disponível em: <http://www.docstoc.com/docs/110376452/(-ATA-VII---Comit%EF%BF%BD-de-Ajudas-T%EF%BF%BDcnicas---CAT>. Acesso em: 06 jul. 2013.

CONDOWORKS. **Elevador com acionamento pela voz.** São Paulo, 2006. Disponível em: < https://condoworks.com.br/noticias/id/425/elevador-com-acionamento-pela-voz>. Acesso em: 17 jul. 2013.

COOK, Albert M.; HUSSEY, Susan M. ***Assistive Technologies: Principles and Practices*.** 3. ed. St. Louis, Missouri: Mosby, 1995.

COSTA, Anna Helena Reali. **Reconhecimento de Fala.** São Paulo, 2008. Disponível em <http://www.lti.pcs.usp.br/pcs2059/aulas/Seminario-G9-doc.pdf>. Acesso em: 12 jun. 2013.

DELGADO, Armando Luiz N. **Linux Básico.** Curitiba, out. 2005. Disponivel em: <http://www.inf.ufpr.br/nicolui/Docs/Livros/LinuxBasico/LinuxBasico.html> acesso em: 17 jul. 2013.

DUBRIN, Andrew J. **Fundamentos do Comportamento Organizacional.** 2. ed. São Paulo: Thomson Learning, 2002.

DYMARSKI, Przemyslaw. ***Hidden Markov Models***, *Theory and Applications*. Rijeka, 2011.

ESTADOS UNIDOS. ***American With Disabilities Act (ADA):*** *Technology-Related Assistance for of 1988.* Estados Unidos, 1988. Disponível em: <http://uscodebeta.house.gov/statutes/1988/1988-100-0407.pdf>. Acesso em: 10 jun. 2013.

FLYNN, Ida M; MCHOES, Ann Mclver. **Introdução aos sistemas operacionais.** São Paulo: Thomsom, 2002.

GALVÃO FILHO, T. A. **A Tecnologia Assistiva: de que se trata?** In: MACHADO, G. J. C.; SOBRAL, M. N. (Orgs.). Conexões: educação, comunicação, inclusão e interculturalidade. 1. ed. Porto Alegre: Redes Editora, p. 207-235, 2009.

GARCIA, Mary Lynn. ***Vulnerability Assessment of Physical Protection System.*** 1. ed. Reino Unido: Butterworth-Heinemann, 2005.

GUEDES, Lucas et al. **O papel social da automação** – automação inclusiva e mais sustentável. In: SEMINÁRIO NACIONAL DE CONSTRUÇÕES SUSTENTÁVEIS. Passo Fundo, 2012.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Censo Demográfico 2010** - Características gerais da população, religião e pessoas com deficiência. Brasil, 2010. Disponível em: <http://www.censo2010.ibge.gov.br>. Acesso em: 15 mar. 2013.

INSTITUTO DE TECNOLOGIA SOCIAL. **Portal Nacional de Tecnologia Assistiva.** Brasil, 2013. Disponível em: <http://www.assistiva.org.br/>. Acesso em: 15 jun. 2013.

INSTITUTO DE TECNOLOGIA SOCIAL. Sociedade Inclusiva e a contribuição da Tecnologia Assistiva**. Revista CONHECIMENTO:** Ponte Para Vida. São Paulo, ano 1, n. 1 e 2, p.38-42, mar. 2007.

KLAUTAU, Aldebaro et al. **Um Reconhecedor de Voz Livre para Português Brasileiro com Interface de Programação.** Universidade Federal do Pará, Belém, 2010.

LABORATÓRIO DE PROCESSAMENTO DE SINAIS. **Reconhecimento de Voz para o Português Brasileiro.** Universidade Federal do Pará, Belém, 2013. Disponível em: <http://www.laps.ufpa.br/falabrasil/descricao.php>. Acesso em: 27 out. 2013.

LEE, Akinobu. ***The Julius Book*.** SourceForge: 2010.

LEITE, Mário. **Técnicas de Programação: uma abordagem moderna.** Rio de Janeiro: Brasport, 2006.

MATHIVANAN, N. ***Microprocessors, Pc Hardware and Intefacing***, Nova Delhi: PHI Learning, 2006.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA. **CHAMADA PÚBLICA MCT/FINEP/Ação Transversal – Tecnologias Assistivas – 09/2005.** Brasil, 2005. Disponível em: <http://www.finep.gov.br//fundos_setoriais/acao_transversal/editais/Chamada_Publica_Acao_Transversal_Tecnologias_Assistivas_09_2005.PDF>. Acesso em: 06 jul. 2013.

MORAES, C. e CASTRUCCI, P. **Engenharia de Automação Industrial.** 2. ed. São Paulo: LTC, 2007.

ORGANIZAÇÃO INTERNACIONAL PARA PADRONIZAÇÃO. **ISO 9999:** Ajudas Técnicas para Pessoas com Deficiência, *2002.*

PATSKO, Luís F. **Tutorial Aplicações, Funcionamento e Utilização de Sensores.** Londrina: Maxwell Bohr, Instrumentação Eletrônica, 2006.

PICONE, Joseph. ***Fundametals of speech recognition: a short course***, *Institute for Signal and Information Processing, Department of Electrical and Computer Engineering, Mississippi State University*. Starkville, 1996. Disponível em <http://speech.tifr.res.in/tutorials/fundamentalOfASR_picone96.pdf>. Acesso em: 2 jun. 2013.

PINHEIRO, José M. S. **Sistemas de Automação.** Jun. 2004. Disponível em <http://www.projetoderedes.com.br/artigos/artigo_sistemas_automacao.php>. Acesso em 16 jul. 2013.

PLANNERER, B. ***An Introduction to Speech Recognition*.** Munique, 2005. Disponível em <http://www.speech-recognition.de/pdf/introSR.pdf>. Acesso em: 3 jun. 2013.

PPA . **Manual do Usuário:** Automatizadores para portoes pivotantes. Garça: PPA, 07 p., 2013. Disponível em: <http://www.netppar.com/Manuais/download/ManualUsuarioPivotante.pdf>. Acesso em: 07 jan. 2014.

PYTHON SOFTWARE FOUNDATION. ***Python Programming Language – Official Website 2013.*** Disponível em <http://www.python.org/about/>. Acesso em: 07 jan. 2014.

RABINER, L. R.; JUANG, B. H. ***An Introduction to Hidden Markov Models*.** Proceedings of the IEEE, v. 77, nº 2. 1989.

RADABAUGH, Mary P. ***Study on the Financing of Assistive Technology Devices of Services for Individuals with Disabilities*** *- A report to the president and the congress of the United State. National Council on Disability*, Estados Unidos, 1993. Disponível em: <http://www.ncddr.org/new/announcements/lrp/fy1999-2003/lrp_techaf.html>. Acesso em: 10 jul. 2013.

RASPBERRY PI FOUNDATION. ***Getting Started Guide.*** Reino Unido: *Raspberry Pi Foundation*, 2012, 13 p.
RASPBIAN. Disponível em: <http://www.raspbian.org/>. Acesso em: 26 jan. 2014

ROCKENBACH, Suzete. **Arquitetura, Automação e Sustentabilidade.** Porto Alegre, 2004.

SASSAKI, Romeu K. **Inclusão: acessibilidade no lazer, trabalho e educação.** Revista Nacional de Reabilitação (Reação), São Paulo, Ano XII, p. 10-16, mar./abr. 2009.

SILVA, Anderson Gomes da. **Reconhecimento de voz para palavras isoladas**, Trabalho de Graduação em Engenharia da Computação, Centro de Informática, Universidade Federal De Pernambuco. Recife, 2009.

SILVA, Ênio; PANTOJA, Marcus; CELIDÔNIO, Jackline; KLAUTAU, Aldebaro. **Modelos de Linguagem N-grama para Reconhecimento de Voz com Grande Vocabulário.** Belém, 2004.

SOMERA, Guilherme. **Treinamento profissional em Java:** aprenda a programar nesta poderosa linguagem. São Paulo: Digerati Books, 2006.

SRIRENGAN, K. ***Understanding UNIX*.** Nova Delhi: PHI Learning, 2006.

TEIXEIRA, Carlos et al. **Elevador controlado por voz.** São Paulo, 2013. Disponível em: < http://prezi.com/no7b4-uovgob/tcc-elevador-controlador-por-voz/>. Acesso em: 20 jul. 2013.

TSCHULENA, Guido R et al. ***Sensors in Household Appliances*.** 5. ed. Weinheim: Wiley-VCH, 2003.

APÊNDICE A - DICIONÁRIO FONÉTICO DIC.TEMP

```
< /s>     [] sil
<s>       [] sil
ativa     a ts i v a
ative     a ts i v e
ative     a ts i v i
ativar    a ts i v a x
aciona    a s i o~ n a
acione    a s i o~ n e
acione    a s i o~ n i
acionar   a s i o n a x
acionar   a s i o~ n a x
liga      l i g a
ligue     l i g e
ligue     l i g i
ligar     l i g a x
anda      a~ d a
ande      a~ d e
ande      a~ d i
andar     a~ d a x
para      p a r a
pare      p a r e
parada    p a r a d a
parou     p a r o w
parar     p a r a x
socorro   s o k o r u
socorro   s o k o r o
socorro   s o k o r u
socorro   s o k o r o
```

APÊNDICE B - PROGRAMA DE RECONHECIMENTO DE VOZ DESENVOLVIDO

```
#Constantes de nivel de confianca e coeficiente de Viterbi
NivelConf = 0.4
CoefViterbi = -10000

#Biblioteca de execucao automatica de aplicativos
import pexpect

#Biblioteca de busca de correspondencia entre strings
import re

#Biblioteca de manipulacao de strings
import string

#Biblioteca das funcoes das GPIO
import RPi.GPIO as GPIO

#Funcao de pausa da biblioteca "time"
from time import sleep

#Configuracoes iniciais das GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#Pino 4: entrada-> sensor
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

#Pino 17: entrada-> fim de curso baixo
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

#Pino 18: entrada-> fim de curso alto
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

#Pino 27: saida-> LED - em movimento
GPIO.setup(27, GPIO.OUT)

#Pino 22: saida-> LED - reconhecimento de voz habilitado
GPIO.setup(22, GPIO.OUT)
#Pino 23: saida-> LED - parado
```

```
GPIO.setup(23, GPIO.OUT)

#Pino 24: saida-> Rele - ativacao
GPIO.setup(24, GPIO.OUT)

#Pino 25: saida-> Rele - parada
GPIO.setup(25, GPIO.OUT)

#Inicializacaso da variavel global de indicacao de Primeiro Ciclo
global PrimCiclo

#Inicializacao das variaveis globais de estado do comando
global ComandoParar
global ComandoMovimento

#Inicializacao das saidas em "False"
GPIO.output(27, False)
GPIO.output(22, False)
GPIO.output(23, False)
GPIO.output(24, True)
GPIO.output(25, True)

#Funcao que busca 'sentence1' no texto de saida do Julius
def AbrirJulius(TextoJulius):
      global PrimCiclo

      match_res = re.match(r'(.*)sentence1(\.*)', TextoJulius, re.S)

      if PrimCiclo==True:
            pass
      elif match_res:
            TestesErro(TextoJulius)

      else:
            print('ERRO!! Problemas ao carregar o Julius')
            pass

#Funcao que busca valores invalidos para as variaveis 'sentence1', 'cmscore1'
e 'score1' no texto de saida do Julius
def TestesErro(TextoJulius):
      GPIO.output(22, False)
```

```
	VetorLinhas = TextoJulius.split("\n")
	for line in VetorLinhas:
		if line.find('sentence1') != -1:
		sentence1 = line
		elif line.find('cmscore1') != -1:
		cmscore1 = line
		elif line.find('score1') != -1:
		score1 = line

	VetorCmscore = cmscore1.split()
	FlagErro = False
	for score in VetorCmscore:
		try:
			score_float = float(score)
		except ValueError:
			continue
		if (score_float < NivelConf):
			FlagErro = True
			print "ERRO!! Nivel de confianca (%.3f) menor que %.3f.
			Tente novamente." % (score_float, NivelConf)

	score1_float = float(score1.split()[1])
	if (score1_float < CoefViterbi):
		FlagErro = True
		print "ERRO!! Coeficiente de Viterbi (%f) menor que %f. Tente
		novamente." % (score1_float, CoefViterbi)
	if (FlagErro== False):
		print sentence1
		print cmscore1
		print score1

		MoverPlat(sentence1)
	else:
		pass

#Funcao que utiliza a plavra de comando para mover a plataforma
def MoverPlat(ComandoFalado):


global ComandoParar
global ComandoMovimento
```

```
VetorStr = ComandoFalado.split()
VetorStrTamanho = len(VetorStr)
if (VetorStrTamanho<2):
      PalavraComando = 'erro'
      print "ERRO!! Comando nao entendido."
elif (VetorStrTamanho>2):
      PalavraComando = 'erro'
      print "ERRO!! Fale uma palavra por vez."
else:
      PalavraComando = VetorStr[1]
      print "OK!! Comando reconhecido: %s" % (PalavraComando)
if     ((PalavraComando=='parar')     or     (PalavraComando=='para')     or
(PalavraComando=='pare')       or       (PalavraComando=='parada')       or
(PalavraComando=='parou') or (PalavraComando=='socorro')):
      if (ComandoMovimento==False):
            print "Plataforma ja esta parada!"
            pass

      elif (ComandoMovimento==True):
            ComandoParar = True
            ComandoMovimento = False
            print "Parando"

            GPIO.output(27, False)
            GPIO.output(23, True)

            GPIO.output(25, False)
            sleep(2)
            GPIO.output(25, True)

elif    ((PalavraComando=='ativa')    or    (PalavraComando=='ative')    or
(PalavraComando=='ativar')      or      (PalavraComando=='aciona')      or
(PalavraComando=='acione')      or      (PalavraComando=='acionar')      or
(PalavraComando=='liga')       or       (PalavraComando=='ligue')       or
(PalavraComando=='ligar')       or       (PalavraComando=='anda')       or
(PalavraComando=='ande')       or       (PalavraComando=='andar')       or
(PalavraComando=='vai')):

      if (ComandoMovimento==False):
            ComandoParar = False
            ComandoMovimento = True
```

```
            print "Em movimento!"
            GPIO.output(23, False)
            GPIO.output(27, True)
            GPIO.output(24, False)
            sleep(2)
            GPIO.output(24, True)

      elif (ComandoMovimento==True):
            print "Plataforma ja esta em movimento!"
            pass

def Julius():

      child.expect('please speak', timeout=None)
      AbrirJulius(child.before)
      print ('Fale...')
      GPIO.output(22, True)

def FimdeCurso(Pino):

      global ComandoParar
      global ComandoMovimento

      ComandoParar = True
      ComandoMovimento = False

      if Pino == 17:
            print "Fim de curso baixo."
      elif Pino == 18:
            print "Fim de curso alto."

      GPIO.output(27, False)
      GPIO.output(23, True)


#Main
if __name__== "__main__":

      global PrimCiclo
      global ComandoParar
      global ComandoMovimento
```

```
PrimCiclo = True
ComandoParar = True
ComandoMovimento = False

if PrimCiclo==True:

      GPIO.output(23, True)
      print ('Bem vindo ao sistema de reconhecimento de voz!!')
      print ('Aguarde, carregando o Julius...')
      child = pexpect.spawn ('julius -C julius.jconf')

GPIO.add_event_detect(17,GPIO.FALLING)
GPIO.add_event_callback(17,FimdeCurso,100)
GPIO.add_event_detect(18,GPIO.FALLING)
GPIO.add_event_callback(18,FimdeCurso,100)

while True:
      if GPIO.input(4):
            try:
                  Julius()

            except KeyboardInterrupt:
                  child.close(force=True)
                  break

            else:
                  print 'Nenhuma pessoa sob a plataforma!'
                  GPIO.output(22, False)
                  sleep(2)

PrimCiclo=False

GPIO.cleanup()
```

APÊNDICE C - INSTRUÇÕES PARA UTILIZAÇÃO DO SISTEMA DE RECONHECIMENTO DE VOZ

As diretrizes para utilização do sistema de reconhecimento de voz serão dadas a seguir.

Passos iniciais: referentes a inicialização do sistema.

1. Conectar o cabo HDMI, ligando o *Raspberry Pi* ao monitor;
2. Conectar a HUB no *Raspberry Pi*;
3. Conectar o teclado, o mouse e o adaptador de áudio à HUB;
4. Conectar o microfone ao adaptador de áudio;
5. Conectar a fonte de alimentação à entrada micro USB do *Raspberry Pi* e ligá-la na tomada;
6. Aguardar a inicialização do sistema operacional *Raspbian.*

Execução do programa de reconhecimento de voz: com o sistema inicializado, através dos comandos listados abaixo, o programa será executado.

7. Logar no sistema utilizando o *login: root* e *password: root*;
8. Declarar a entrada do microfone como entrada de áudio principal:

```
export ALSADEV="plughw:1,0"
```

9. Entrar na pasta que contém o arquivo de programa a ser executado:

```
cd root/coruja_jlapsapi
```

10. Executar o programa de reconhecimento de voz desenvolvido:

```
python programa.py
```

11. Sistema pronto pra uso e aguardando comando de voz do usuário!

Uma vez que o programa esteja em execução, os periféricos de programação não são mais necessários e podem ser desconectados do sistema, sendo eles: o monitor, o teclado e o mouse.

A Figura 32 mostra a tela de inicialização do RPi após dados os comandos:

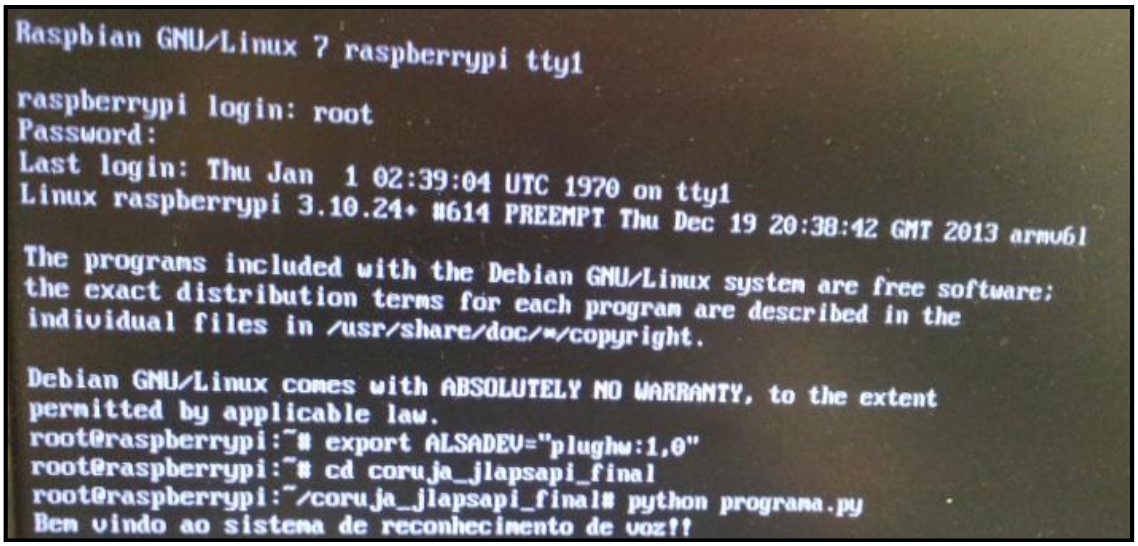


Figura 32 - Tela de inicialização do sistema de reconhecimento de voz
Fonte: Autoria própria.

ANEXO A - ARQUIVO JULIUS.JCONF

```
#############################################################################
# Julius Conf File
#############################################################################

-lv 3000         # level threshold (0-32767)
-zc 150            # zero-cross threshold (times in sec.)
-headmargin 600      # head silence margin (msec)
-tailmargin 1000      # tail silence margin (msec)
-rejectshort 50         # reject shorter input (msec)

-realtime         # force real-time processing
#-norealtime         # force non real-time processing



# Specify short pause model name to be treated as special
-spmodel "sp"   # HMM model name
#-iwspword


# Context-dependency handling will be enabled according to the model type.
# Try below if julius wrongly detect the type of hmmdefs
#
#-no_ccd    # disable context-dependency handling
#-force_ccd     # enable context-dependency handling

#-htkconf ../../confs/edaz_mod.conf


#-looktrellis

# LM weights and word insertion penalties
#-lmp 15.0 10.0
#-lmp2 15.0 10.0
#-transp -20.0

# Envelope beam width (number of hypothesis) - deixa o decoder bem mais
lento !!
#-b 2000
#-b2 200
#-sb 300

# The number of candidates Julius tries to find.
#-n 1

## If julius go wrong with checking parameter type, try below.
##
#-notypecheck
#

## (PTM/triphone) switch computation method of IWCD on 1st pass
##
#-iwcd1 best 5  # assign average of N-best likelihood of the same context
#-iwcd1 max # assign maximum likelihood of the same context
#-iwcd1 avg # assign average likelihood of the same context (default)
```

```
#### Gaussian Pruning ####
## Number of mixtures to select in a mixture pdf.
## This default value is optimized for IPA99's PTM,
## with 64 Gaussians per codebook
#-tmix 2

## Select Gaussian pruning algorithm
## defulat: beam (standard setting), safe (others)
#-gprune safe   # safe pruning, accurate but slow
#-gprune heuristic   # heuristic pruning
#-gprune beam   # beam pruning, fast but sensitive
#-gprune none   # no pruning


#### Search Parameters ####

-force_ccd     # enable context-dependency handling
-lmp 15.0 10.0
-lmp2 15.0 10.0
-b 2000
-b2 200
-sb 300


#-b 400                 # beam width on 1st pass (#nodes) for monophone
#-b 800                 # beam width on 1st pass (#nodes) for triphone,PTM
#-b 1000                # beam width on 1st pass (#nodes) for
triphone,PTM,engine=v2.1
#-s 50        # Stack size
#-b2 200                # beam width on 2nd pass (#words)
#-sb 200.0     # score beam envelope threshold
#-s 500                 # hypotheses stack size on 2nd pass (#hypo)
#-m 2000                # hypotheses overflow threshold (#hypo)
#-lookuprange 5         # lookup range for word expansion (#frame)
#-n 1                   # num of sentences to find (#sentence)
#-n 10                  #   (default for 'standard' configuration)
#-output 1            # num of found sentences to output (#sentence)
#-looktrellis   # search within only backtrellis words


## For insertion of context-free short-term inter-word pauses between words
##  (multi-path version only)
##
-iwsp     # append a skippable sp model at all word ends
#-iwsppenalty -5.0   # transition penalty for the appenede sp models


######################################################################
#### Speech Input Source
######################################################################
## select one (default: mfcfile)
#-input mfcfile         # MFCC file in HTK parameter file format
#-input rawfile         # raw wavefile (auto-detect format)
                        # WAV(16bit) or
                        # RAW(16bit(signed short),mono,big-endian)
                       # AIFF,AU (with libsndfile extension)
 # other than 16kHz, sampling rate should be specified
 # by "-smpFreq" option
#-input pulseaudio
-input mic             # direct microphone input
 # device name can be specified via env. val. "AUDIODEV"
```

```
#-input netaudio -NA host:0    # direct input from DatLink(NetAudio) host
#-input adinnet -adport portnum # via adinnet network client
#-input stdin   # from standard tty input (pipe)

#-filelist filename  # specify file list to be recognized in batch mode

#-nostrip      # switch OFF dropping of invalid input segment.

-zmeanframe                          # (default: strip off invalid segment (0
sequence etc.)
-zmean        # enable DC offset removal (invalid for mfcfile input)

#########################################################################
#### Acoustic Analysis
#########################################################################
-smpFreq 20050   # sampling rate (Hz)
#-smpPeriod 227      # sampling period (ns) (= 10000000 / smpFreq)
#-fsize 400    # window size (samples)
#-fshift 160   # frame shift (samples)
#-delwin 2     # delta window (frames)
#-hifreq -1    # cut-off hi frequency (Hz) (-1: disable)
#-lofreq -1    # cut-off low frequency (Hz) (-1: disable)
#-cmnsave filename   # save CMN param to file (update per input)
#-cmnload filename   # load initial CMN param from file on startup

#### Acoustic HMM file ####
-AM lapsam
# Hmm model
-h ./LaPSAM1.5.am.bin
-iwcd1 best 5  # assign average of N-best likelihood of the same context
-gprune safe      # safe pruning, accurate but slow

# triphone model needs HMMList that maps logical triphone to physical ones.
-hlist ./LaPSAM1.5.tiedlist

# Specify short pause model name to be treated as special
-spmodel "sp"     # HMM model name
#-iwspword

#### Language Model ####

-LM dicSr
-d ./LaPSLM1.5.lm.bin
-v ./dic.temp

#-LM grammar
#-w sample.dict

-SR dicSr lapsam dicSr
#-SR gramSr lapsam grammar
#-SR sample.dict lapsam
```